Università degli Studi di Padova

Dipartimento di Matematica

Corso di Laurea Magistrale in Informatica

Solving Signaling Storms in LTE Networks: a Software-Defined Cellular Architecture

Author:
Matteo Pozza

Supervisors:
Claudio Enrico Palazzi
Sasu Tarkoma

June 23, 2016

Academic year 2015/2016

Contents

List of Figures

List of Tables

Abstract

The LTE network infrastructure is composed by monolithic devices that carry out a convoluted set of tasks in a vendor-specific manner. Therefore, LTE networks are largely inflexible, and consequently unable to adapt to a constantly increasing number of mobile subscribers and the changeable usage pattern of the Internet service. In fact, current LTE networks are affected by signaling storms, which come from the inability to reduce the number of signals exchanged among the infrastructural elements of the network when the number of subscribers' requests grows. In this work, we propose a software-defined cellular architecture, whose logical entities can be mapped to an arbitrary number of physical devices, allowing different implementations depending on the specific use case. In particular, we show that the proposed model actually mitigates the impact of signaling storms, as it can be tailored to reduce significantly the number of signals flowing in the network during the occurrences of the most frequent network events.

Chapter 1

Introduction

Since 2004, the 3rd Generation Partnership Project (3GPP) has been working on the standardization of a new technology for mobile devices, called Long Term Evolution (LTE) [26]. The LTE standard is the outcome of a complex attempt to supply high-speed Internet connection to mobile users taking into account retrocompatibility with older cellular standards at the same time. In order to be adopted, the technology requires enhancements in the mobile devices, as well as modifications in the infrastructure of the mobile operators proving the services. The resulting architecture is composed by several networking elements that exchange signals in order to provide Internet connectivity and legacy cellular services to mobile subscribers. Since a high-speed Internet service requires a significant amount of battery power, a permanent connection through an LTE network would lead mobile devices to run out of energy in few hours. For this reason, LTE networks provide also some mechanisms that allow users' devices to transit from the normal active state to a power-saving one, and vice-versa. Limiting the perspective on these few aims, the LTE standard has fully reached its original objectives.

Nevertheless, the main issue of the LTE architecture is its intrinsic inflexibility, which comes from the way in which its entities have been designed and are implemented. In fact, each network element of the architecture is in charge of carrying out a convoluted set of tasks, whose aims are significantly different [73]: some of them deal with the forwarding of data packets, some others with the management of the data flows, some others with storing and retrieval of subscribers' information. This leads the network elements to have imprecise roles, with a paradoxical lack of standardization in their behaviour. Therefore, the producers of network infrastructural components implement the entities in a vendor-specific manner, which implies that mobile operators are affected by vendor-locking, as components of one vendor may be incompatible with the ones of another vendor. Moreover, given the intricate roles of the entities, vendors tend to limit the customization opportunities provided on each device [117]. This leads to two disadvantages: first, it increases the vendor-locking phenomenon, as non-customizable devices are even less likely to work with other non-customizable devices, and second, it does not allow mobile operators to try the roll-out of new services, but they have to wait devices' vendors to implement them.

The inflexibility of the LTE architecture makes it unable to change together with the mutable world of mobile subscribers. In fact, LTE networks should be able to adapt to the number of subscribers, as well as to the way in which the services provided are used by subscribers. Instead, the 3GPP standard does not mandate any mechanism

that allows LTE networks to scale with their load: this automatically implies that this inability manifests somehow as soon as they face some issues in handling subscribers' requests. Indeed, nowadays LTE networks are experiencing signaling storms, meaning that they are spending a prohibitive amount of time and physical resources just for handling the signals exchanged internally by their entities. This happens because every time a network event occurs (*e.g.* user initiating a session, device going idle) a procedure is triggered among the entities of the LTE network (*e.g.* initial attach, handover), and each procedure implies a considerable amount of signals that does not change with the number of simultaneous users. Therefore, a growing number of LTE subscribers naturally implies an increase in these signals, which LTE networks hardly manage. Moreover, the constant need of small keep-alive messages typical of popular mobile applications (*e.g.* WhatsApp, Messenger) leads mobile phones to continuously turn from idle to active state and vice-versa [67]. This implies an increasing number of per-subscriber signals, as their amount does not depend on the size of data to upload or download. As the number of mobile subscribers is expected to grow in the next years [61], the signaling storms issue has to be solved as soon as possible.

We argue that signaling storms can be avoided through an injection of flexibility inside the current LTE architecture. The first step is the identification of homogeneous sets of tasks: from the convoluted roles of the LTE entities, we group tasks participating to the same aims. Once the categories of tasks are defined, we re-design the roles of the devices as sets of tasks belonging to the same category: this simplification of the roles leads their implementation to be unambiguous. Finally, we can coalesce entities belonging to the same category in order to reduce the number of signals exchanged during each procedure.

Nevertheless, there is a need of a networking paradigm that tells us under which categories we should group the tasks carried out by LTE entities. Software-Defined Networking (SDN) is an approach used to organize networks which mandates a clear separation between the transmission of data, called data plane, from its management, called control plane. The benefits brought by this paradigm are programmability and easier administration of the network [34], and therefore we decided to leverage these abstractions to perform our grouping. Secondarily, we need also an approach that allows an easy shifting of tasks from one entity to another one. In this context, Network Function Virtualization (NFV) is another paradigm that mandates the execution of network functions on a virtual machine that runs on commodity hardware. The paradigms brings easiness in coding, moving, and testing of new services [69], and therefore it is perfectly suitable to meet our requirements.

The outcome of our work is a software-defined cellular architecture functionally equivalent to the current LTE one. Therefore, it provides the services needed by LTE-enabled devices, bringing the benefits of SDN and NFV at the same time. In particular, we show that the provided model is able to reduce significantly the impact of signaling storms in the load handled by the network. Moreover, the flexibility of the proposed architecture makes it suitable to meet the requirements coming from different verticals. In this context, the signaling storm avoidance is just one of the many properties for which the architecture can be customized. In fact, the modularity of the model allows mobile operators to tailor the LTE network on their specific necessities. On this line, we propose a single-device implementation of our model that we believe to be the basic building block of the future LTE networks.

1.1 Contributions

We can summarize the contributions brought by our work in the following list:

- We provide extensive documentation that discusses, analyzes, and evaluates the current LTE architecture. In particular, we have focused our understanding on four procedures that are frequently triggered in LTE networks. These are: initial attach, transition from active state to idle state, transition from idle state to active state (both triggered by the network and by the user equipment itself), and handover (both X2 and S1). We describe which are the state variables involved in these procedures, the role of each one of them, and when they are created, transmitted, and destroyed. We report the list of tasks that are carried out by an LTE network during these procedures, and we explain which are the signals exchanged when the procedures are triggered. We also provide slides that can result in a didactic graphical tool to easily explain the considered procedures [94].
- We present a software-defined model for LTE networks that do not require any modification of users' devices and that can be implemented through different configurations. Depending on the implementation, the advantages brought by the model are different: therefore, the model allows mobile operators to tailor the network to their specific needs. In particular, one of these implementations brings a reduction in the signals of about 50% for handling some network events. Therefore, our proposed model is able to contrast concretely the impact of signaling storms in the cellular architecture. We show how the model has been conceived, going through all the phases of the whole refactoring process, and providing graphical tools that allow to understand how the procedures change accordingly [101, 108, 115].
- We present our idea of network-in-a-box, an implementation of the proposed model in a single physical device. The device is able to provide a fully-fledged LTE network, as well as to collaborate with other devices for the same aim. It represents the fundamental brick of the future cellular network, as it allows an organic growth of the network and it opens new opportunities for fault tolerance mechanisms. The practicality of the device makes LTE networks suitable for a completely new set of scenarios, as organization of rescue operations or Internet service supplying in remote locations.

1.2 Outline

The rest of the document is organized as follows. Chapter 2 provides a detailed discussion about the current LTE architecture, its issues and the two emergent network paradigms, Software-Defined Networking and Network Function Virtualization. Chapter 3 reports the technical details about the variables composing the state of the network, the procedures triggered by some network events, and the tasks carried out by the entities of each LTE network. Chapters 4, 5 and 6 describe the refactoring steps that we have performed in order to obtain the proposed model. Chapter 7 reports the analysis process adopted to evaluate the model, showing the related results. A discussion about similar works that have been performed on the same topic is reported in Chapter 8. Finally, Chapter 9 summarizes the outcomes of the work and provides insights for improvements.

Chapter 2

State of the Art

In this chapter, we provide the background knowledge that motivates the refactoring process that we have carried out. First, we present the current LTE architecture, explaining the functions of each one of its components. Then, we focus on the issues affecting the current LTE architecture, and on signaling storms in particular. In this section, we also discuss which are the causes of this phenomenon. Finally, we describe the theory we are going to leverage in order to solve the presented issues. We also report a brief discussion on similar attempts in the same research field.

2.1 The Current LTE Architecture

We start providing an overview on the current LTE architecture. From now on, we will use LTE, and not 4G, to describe the kind of networks we are referring to. In fact, as reported in [59], 4G matches a set of requirements, including minimum bandwidth, maximum downlink and uplink connection speed values, and so on. Instead, LTE is the outcome of a standardization process performed by 3GPP whose aim was to reach the 4G requirements. Actually, the final achievements of LTE are lower than the 4G requirements. Nevertheless, it represents an evolution from the previous HSPA protocols, which were labelled as 3.5G, and therefore the research community has labelled it as 4G.

Figure 2.1 depicts a schematic view of the current LTE architecture, mainly inspired by our understanding from [16, 48]. First, we can recognize four groups of entities: the LTE-enabled devices, in orange; the Radio Access Network (RAN), in yellow; the core network, in pink; the Internet, in gray. The LTE-enabled devices are all the devices that use the services provided by the LTE network, like smartphones and tablets, for example. In the figure, they are labelled as User Equipment (UE). The RAN comprises all the base stations of the network, so its primary aim is the communication with the LTE-enabled devices. The core network takes care about directing the traffic from and to the user equipments, acting as a data bridge between the RAN and the Internet.

Each UE connects to a base station, called eNB, through the LTE-Uu interface. The eNB is connected in turn to two core entities: the Mobile Management Entity (MME) and the Serving Gateway (S-GW). The MME, which uses the S1-MME interface to communicate with the eNB, takes care about the establishment and release of data bearers from the UE to the Internet. It also executes the procedures that ensure a secure communication channel between the UE and the cellular network. Instead,

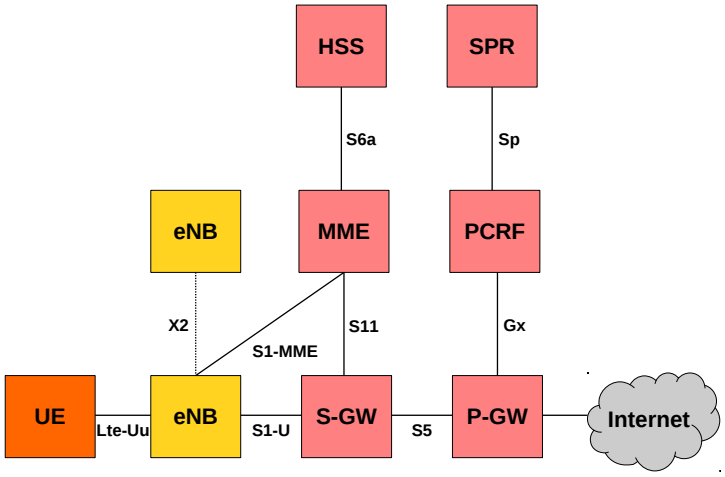


Figure 2.1: **The current LTE architecture.** *The entities in yellow belong to the Radio Access Network (RAN), while the pink ones belong to the Core Network.*

the S-GW acts as a mobility anchor during handover events that involve base stations linked to it. It takes care about buffering the downlink packets when the UE cannot receive them as well. The S-GW communicates with the eNB through the S1-U interface.

The Home Subscriber Server (HSS) is a database-like structure that stores customer-specific values. It is linked to the MME through the S6a interface [18], and the information it stores are QoS parameters and values used during security procedures, for example. Instead, the S-GW is linked to another gateway, which is the PDN Gateway (P-GW), through the S5 interface. The P-GW consists in the last gateway before reaching the Internet. It takes care about assigning IP addresses to the UEs and enforcing the QoS rules received from the Policy and Charging Rules Function (PCRF). In fact, the PCRF creates the QoS rules that shape the traffic in the cellular network [20]. It is connected to the P-GW through the Gx interface. In order to generate the QoS rules, the PCRF retrieves other customer-specific values from another database-like structure, the Subscriber Profile Repository (SPR). The SPR communicates with the PCRF through the Sp interface.

Please notice that two eNBs may be directly linked as well. In this context, the eNBs use an X2 interface in order to communicate. The interface between eNBs becomes useful during handovers, as the eNBs can organize the greatest part of the procedure directly among themselves, without the need of other external entities.

2.2 The Signaling Storms Phenomenon

The current LTE architecture is the output of a complex attempt to combine the requirements of 4G and the functions for backward compatibility. Nevertheless, it has not been designed taking into account the usage trends of mobile users and their increasing number.

Every time a network event occurs, like a UE attaching to the network or performing an handover, the LTE network reacts exchanging a certain number of signals among its components. This amount of signals varies with the network event to handle, but generally can be considered non-negligible. Therefore, a growing number of mobile subscribers leads inevitably to a significant increase in the number of signals exchanged among the LTE entities, which may become difficult to handle properly. As reported in [120, 61], there will be around 24 billions of devices connected to the Internet in 2020, and 3.7 billions of LTE subscriptions. This increased amount of mobile users will lead to an explosion of signals in the LTE networks: in fact, a linear growth in the number of users implies an exponential growth in the number of signals to be exchanged among the entities of the LTE architecture [38, 121].

The evolution in the usage of the services provided by LTE networks has lead to a worsening of the situation. In fact, the current trend of accessing Internet services using mobile phones is through short and frequent connections. This means that a UE establishes a connection and releases it after a while, instead of keeping the connection active, as performed by laptops for example [87]. This happens mainly for two reasons: battery saving, and keep-alive behaviour of installed applications.

In order to extend battery life as much as possible, mobile devices alternate between idle and active state. This happens because HD screens and radio assets ask for a significant (and increasing) amount of battery power. Every time a UE performs a transition between the two states, a signals sequence is triggered in the LTE network. Therefore, the growth in the number of LTE subscribers leads to a consequent increase in the number of idle-active transitions, which implies an increase in the signaling in turn [29, 39, 67, 87].

Moreover, nowadays the post popular mobile applications, like WhatsApp or Facebook, require an ad-hoc Internet connection at periodic intervals in order to refresh their contents and download updates in their status. This behaviour is characterized by heartbeat messages with a small temporal gap between each other. The absence of coordination among the several apps lead the mobile device to continuously establish and release connections, increasing further the amount of signals exchanged by the entities of the LTE networks. In fact, the data carried by the heartbeat messages is small, but the number of signals exchanged internally by the LTE network is not related to the amount of data to download or upload. Therefore, the signaling traffic is increasing by 30% - 50% over the user data traffic [29, 33]. Many studies have identified this keep-alive behaviour as one of the main sources of the impressive increase in signaling [29, 39, 67, 84, 87, 121].

Figure 2.2 shows how the average number of signals sequences triggered by a single subscriber has grown in the last years. The factors we have explained previously have lead to this glaring increase, and the number of signals is expected to continue to grow in the years to come as well. The amount is so huge that the components of these networks are being pushed to their limits, and since the amount is expected to increase, a solution to the problem has to be found. Given the importance, the phenomenon has been labelled with a specific name, which is signaling storm.

A straightforward solution may be to increase the number of network elements providing the same service, a technique which is called horizontal scaling. Unfortunately, a linear growth in the number of entities leads to a quadratic number of connections required among them [87]. Therefore, this approach is not only expensive, but it leads also to an increase in the number of signals, as the entities providing the same services have to exchange messages for consistency purposes.

We argue that the source of this problem consists in the inflexibility affecting the

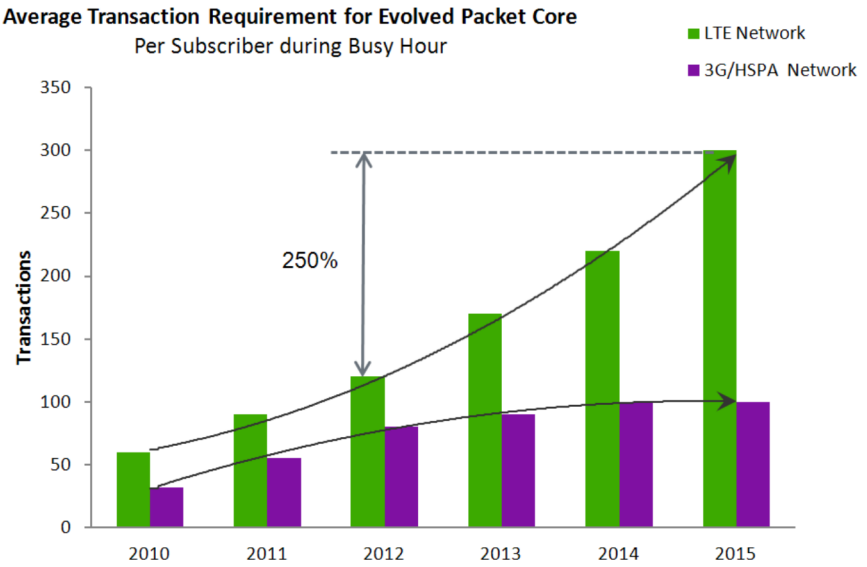


Figure 2.2: **Increase in the signaling during the last years.** *In the chart, a transaction matches a network event to handle, and therefore it corresponds to a sequence of signals in the LTE network. The chart is picked from [84].*

current LTE architecture. In fact, the elements composing LTE networks are usually considered as black-boxes by mobile operators: they carry out a huge and fixed set of tasks, without providing any customization possibility. Mobile operators are forced to configure the network elements in a fixed, pre-defined, vendor-specific manner, and usually they have no possibilities of upgrading them or tailoring their behaviour to the mobile operator's wills [117]. In this context, the network architecture cannot evolve to meet the emergent requirements, because it is inherently immutable. This phenomenon is known as ossification of the current LTE architecture.

This idea has just started to be explored. In fact, the recent work of Qazi *et al.* [116] identifies the inflexibility of the current LTE architecture as the main source of resource wastages as well as of signaling storms. In this scenario, the current LTE architecture has to evolve in something more elastic, which can easily adapt to the customers' needs. Nevertheless, this transition cannot modify the way in which user equipment access to cellular networks, as a substitution or upgrade of all of the widespread LTE-enabled devices is undesirable as well as unfeasible. We need to change the internal structure of the LTE architecture, keeping the same external interface to the users. Therefore, we need a refactoring process.

2.3 Software-Defined Networking and Network Function Virtualization

From the previous section, we know that, in order to solve the signaling storm issue, the refactoring process has to inject flexibility in the current LTE architecture. In order to achieve this, we base our refactoring process on two ideas. First, we want to organize the convoluted set of tasks assigned to the LTE elements, such that homogeneous

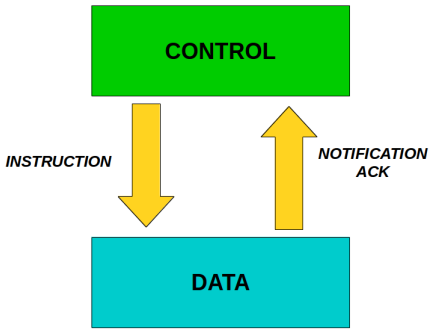


Figure 2.3: **Interface between control and data planes.** *The data plane provides notifications, acknowledgements, and statistics to the control plane, which takes the management decisions accordingly and instructs the data plane consequently.*

tasks are carried out by the same logical entity. With homogeneous tasks we mean tasks that participate to the same aims, using the same information: we say that homogeneous tasks belong to the same category. This organization inherently implies separating the tasks of each element by category, and then merging the tasks belonging to the same category in the same logical entity. In this way, we gain entities whose roles are well-defined. Second, we want to provide an architecture that allows the insertion and the removal of functions from these entities. This would allow to modify the behaviour of the network depending on the mobile operator's wills. Therefore, we leverage two network paradigms that embody these ideas, which are Software-Defined Networking (SDN) and Network Function Virtualization (NFV) [62].

Software-Defined Networking mandates a clear separation between the control plane and the data plane in the network. The control plane corresponds to the set of functions that takes care about the management of the network, like redirection of the traffic or limitation of the data rate, for example. Instead, the data plane matches all the functions that allow the actual data transmission, as execution of the lower-level protocols or CRC checks.

Figure 2.3 depicts the uniform interface between control and data planes. Being related with the actual communication channel, the data plane informs the control plane about statistics on the data flows. These may include average speed, average packet loss, and any other related information. Moreover, it provides the high-level acknowledgements as well as the notifications about networking issues (*e.g.* destination unreachable). The control plane uses the received information to take decisions about the management of the network. As an example, a too high average packet loss may trigger the control plane to the decision of changing the data path to a more reliable set of links. Therefore, the control plane instructs the data plane according to the decision performed. The data plane will react changing according to the instructions received, and it will provide the statistics to the control plane again, continuing the loop.

The main advantage brought by SDN is easiness in management. In fact, all the

intelligence of the network is shifted to the control plane, while the behaviour of the data plane is simple: obey to the instructions, and collect statistics. The dumbness of the data plane implies that its peripherals have an easy and standard behaviour, rather than the complex and vendor-specific one we were describing about the LTE networks' components. These characteristics allow the data plane to be composed by cheap and interchangeable devices. In fact, since the interface between the control and data planes is uniform, the control plane is the same regardless of the components of the data plane.

Network Function Virtualization mandates the insertion of an abstraction layer between the physical components of the network and the functions that the networking hardware is supposed to execute. Instead of having network functions which are tailored to the hardware on which they have to run, this paradigm asks for the presence of a virtual machine that intermediates between the actual device and the software of the network function. The virtual machine exposes a standard interface to the network function's software, while it takes care about the conversion between hardware-specific language and the language of the standard interface provided. In this way, network functions become portable and combinable, as all the burden is shifted on the virtual machine. The network function firewall written for the gateway A is perfectly suitable for any gateway B as well.

We exemplify the paradigm though a paragon with the Java programming language. Java has been conceived as the answer to the problem that C++ programs were tailored for the specific operating system. In fact, in order to deploy a software for several operating systems, most of the times the code needed to be partially refactored to meet the requirements of each system. Therefore, Java comes with the Java Virtual Machine (JVM), which consists in an abstraction layer between the operating system and the Java code. The JVM is tailored specifically for each operating system, but the interface offered to Java programmers is standard. This allows the programmers to write a single version of their code, since the same Java code will run with the same behaviour on each operating system. Moreover, the JVM can easily merge Java code coming from different sources, thanks to the standard interface offered to Java programmers. Given this picture, the idea of network function virtualization is exactly the same: allow a single version of a network function to run on every network device, and allow the free composition of different network functions as well.

When SDN and NFV are used together, the resulting picture is depicted in Figure 2.4. The abstraction layer that NFV requires is represented by the Network Operating System (NOS): it takes care about the communication with the physical network elements, while providing a uniform interface to the network functions. The physical devices are software-defined, and therefore the relationship that holds between them and the NOS is the same described in Figure 2.3. Please notice that the decisions are actually taken by the network functions, which represent the real intelligence of the network: in this context, the NOS acts only as intermediary between the physical layer and the smart layer. The network functions communicate with the NOS through a northbound interface, which corresponds to the Java language in the previous comparison. An example of northbound interface is Frenetic [63], a programming language that allows to write network functions. Instead, the interface between the NOS and the software-defined devices is called southbound interface. The most common southbound interface is OpenFlow [75], a standard that specifies which are the commands that the control plane can issue to the data plane.

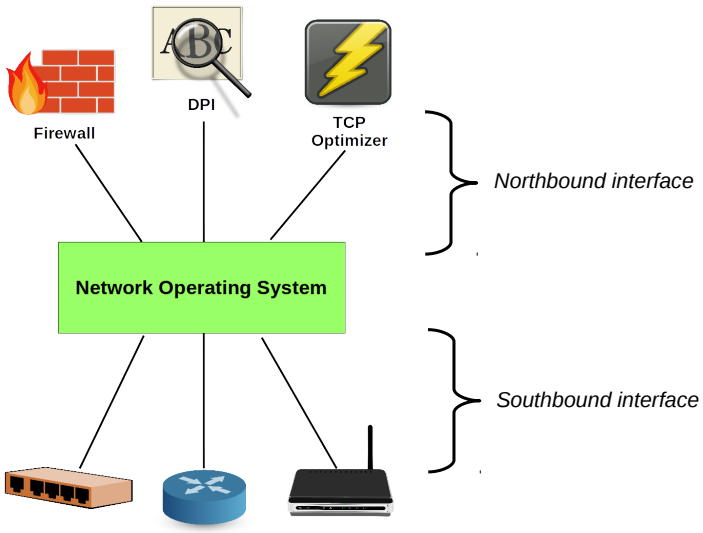


Figure 2.4: **Merging together SDN and NFV.** *The software-defined hardware devices communicate with the Network Operating System (NOS) through a southbound interface (e.g. OpenFlow). The different network functions use a northbound interface (e.g. Frenetic) in order to execute on top of the NOS.*

2.4 SDN and NFV on LTE Networks

At this point, a natural question is how to contextualize the principles of SDN and NFV in the LTE scenario. Many works have already explored the application of these two paradigms [37, 78], and they have implicitly agreed on a basic, common model.

The physical infrastructure that takes care about the data flows between the UEs and the Internet belongs to the data plane. Its components are base stations, gateways and switches, whose main common feature is the dumbness. In fact, they provide a software-defined interface to the control plane, and they maintain less state as possible. Instead, the control plane is moved to the cloud, such that it can scale dynamically with the load in the network. The actual decisions are taken by the installed network functions, which mainly corresponds to the tasks carried out by MME and PCRF. Moreover, the mobile operators can install additional network functions in order to improve the services provided: for example, a load balancer to steer the traffic properly in the network, or a TCP optimizer to improve the data flows of premium subscribers.

Among the advantages, SDN and NFV bring load optimization and energy saving [34]. In fact, the hardware can be set up and torn down dynamically thanks to the newly introduced virtualization layer. For example, we can increase the computational capacity when we are experiencing busy hours, as well as reduce it during the night. In the same way, we can set up temporary base stations when experiencing flash crowds, like during sport events. This elasticity allows to avoid resource wastages, and therefore it allows to optimize energy consumption. Other advantages are rapid service innovation and dynamic service chaining [69, 35]. In fact, as pointed out previously, a mobile operators can improve the services offered simply installing new

network functions, which can be bought from a dedicated marketplace [78], as well as coded by the mobile operator itself to test innovative ideas. Moreover it is easy to link the different services in chain, using the output from one service as input for the following one. This is allowed by the flexibility introduced by the virtualization layer, as the functionalities provided by the network elements can be easily customized. As an example, a mobile operator could provide a firewall service on data flows for all subscribers, an additional service of Deep Packet Inspection for premium subscribers, and an optimizer as a service for loyal subscribers.

Nevertheless, SDN and NFV comes with drawbacks as well. Unquestionably the virtualization layer adds computational overheads, which lead to a degradation of the performance. Moreover, the split between control and data planes implies communication links between them. This may represent an issue both from the implementation perspective, as the additional links could be expensive in the cellular network architecture, and from the processing delay perspective, since the network elements cannot handle the events locally anymore, but they have to contact the control plane instead.

Chapter 3

The Current LTE Architecture

In this chapter we go through the details of the current LTE architecture. In fact, we performed a refactoring process that modifies some very specific features, as the typology of signals used and the information carried by them. Therefore, here we provide an extensive explanation of the different aspects we have touched in our research work. In particular, we focus on four aspects of current LTE networks: the state variables, the considered network events, the signaling load, and the tasks carried out.

3.1 The State Variables

Every time a subscriber interacts with an LTE network, information are created, exchanged, and destroyed in the components of the network. These information represent the state of the UE of the subscriber that is interacting with the network: they are its identifiers, its status (*e.g.* active, idle), its subscription details, and so on. For this reason, these information are called state variables.

The amount of information that each network event involves is enormous. For this reason, it is useful to organize them in categories. The work of Lindholm *et al.* [74] proposes a subdivision in 5 categories:

- Device identification, for all the variables involved with the identity of the subscriber and of its UE;
- Location, for the variables that deal with the position of the UE, as the cell to which it is attached;
- Radio session state, which includes all the information related with the radio communication between UE and eNB;
- Control plane connection state, for the variables related with the management of the data tunnel between UE and the Internet, like security information;
- User plane connection state, for the details of the data tunnel, as QoS parameters, IP of the UE, and so on.

We have considered a slightly different subdivision: while we keep the first two categories, we merge and split differently the last three. The subdivision comes from the

many documents we have been studying during the work [45, 47, 46, 52, 49, 64], in which further details about the role of each state variable are available as well. The adopted subdivision follows:

- UE identifiers, as the previous device identification category;
- Location information, as the previous location category;
- Security information, which includes both radio-level security (AS security) and network-level security (NAS security);
- Session information, for all the variables related with the setting of the data bearers (QoS parameters, data tunnel identifiers, and so on).

As the amount of state variables LTE networks deal with is really huge, considering all of them is cumbersome. Therefore, we have focused just on a significant amount of them. In particular, we have omitted the low-level ones, just considering the most representative ones. For simplicity reasons, some smaller variables have been grouped in a single comprehensive state variable. In the following, we describe all the state variables we consider in our work.

3.1.1 UE identifiers

Here we describe all the state variables that are related with the identification of the UE and of its corresponding subscriber.

- IMSI: the permanent and universal identifier of the subscriber (*i.e.* the owner of the UE). Please note that IMSI includes also the PLMN ID, a identifier for the network (mobile operator and country);
- GUTI: the temporary universal identifier of the subscriber, used in place of the IMSI for security reasons;
- UE IP Address: the IP assigned by the P-GW to the UE, in order to allow an IP-oriented communication with the Internet;
- C-RNTI: the identifier assigned by the eNB to a UE in order to identify it in a cell;
- eNB S1AP UE ID: used to distinguish control plane communications over the same S1 link between eNB and MME in the eNB;
- MME S1AP UE ID: used to distinguish control plane communications over the same S1 link between eNB and MME in the MME.

We briefly explain the last two variables. Since we have a single link between eNB and MME, we need a mechanism that allows to multiplex several user-specific communications on the same link. To this end, we have eNB S1AP UE ID and MME S1AP UE ID. The first one is allocated by the eNB and it is sent to the MME, which can use it as destination for the commands to the eNB and also as sender when checking the origin of a packet coming from the eNB. Instead, the second one is allocated by the MME and it is sent to the eNB for the same identical purposes of the first one.

3.1.2 Location information

Here we list the information that are related with the geographical position of the UE.

- ECGI: permanent and universal identifier of the cell to which the UE is attached (or is attaching);
- TAI: permanent and universal identifier of the tracking area (i.e. set of cells, possibly of different eNBs) to which the ECGI belongs;
- TAI List: list of the TAIs of a set of neighbour TAs;
- TAU Timer Value: the time that has to pass after each Tracking Area Update (TAU);
- MME ID: permanent and universal identifier of a MME.

Usually, a eNB handles more cells at the same time, which are uniquely identified through an ECGI. The cells are organized in groups, which are called Tracking Areas (TAs). Each TA has a unique identifier, the Tracking Area Identifier (TAI).

The utility of the TA comes from the wakening up of the UE. In fact, if the UE is idle and it has to be waken up, its current cell is unknown to the LTE network, but the TA is known instead. Therefore, the LTE network limits the broadcast of the wake-up message to the cells belonging to the tracking area, being sure that the UE will be reached eventually.

The TAs may be organized in higher-level groups, and each group matches a list of TAs. In this case, we have a TAI list as well, which is the list with all the TAI of the TAs belonging to the same group. A UE which travels inside the coverage area of the TAs in the TAI list avoids to send continuous updates about its movements in the different TAs, bringing advantages both in signaling in the LTE architecture and in energy consumption. The obvious drawback is that during transitions from idle to active state, the LTE network has to extend the diffusion of the wake-up message to all the TAs in the TAI list, instead of a single TA.

Finally, the UE has to perform an update about its position after a certain timeout, regardless TA or TAI list. The amount of time the timer is set matches the TAU Timer Value.

3.1.3 Security information

The information we have considered related to the security procedures follow.

- LTE K: LTE security key, stored both in the UE and in the HSS;
- UE Network Capability: the set of security algorithms that UE supports;
- RAND: random value used during mutual authentication;
- AUTN: authentication token used during mutual authentication;
- XRES: value used during mutual authentication;
- RES: same as XRES, but it is generated by the UE (instead of being generated by HSS);
- K_{ASME}: top-level key used during mutual authentication;

- KSI_{ASME}: an index for K_{ASME};
- NAS Security Algorithm: the algorithm(s) used for ciphering and integrity between UE and MME;
- UE Security Algorithm: the algorithm(s) used by the eNB to create security keys;
- AS Security Algorithm: the algorithm(s) used for cyphering and integrity between UE and eNB;
- K_{eNB}: main key used for secure communication between UE and eNB.

From a high-level perspective, we can recognize three security procedures that are carried out by the LTE network: authentication, NAS security, and AS security.

The authentication is the process that allows to make UE and LTE network aware of each other. The LTE network has to verify that the UE is really the one believed, and same holds for the UE: it wants to be sure that the LTE network is the desired one.

To establish a secure communication channel between the UE and the LTE network, a NAS security procedure is carried out. After this, UE and LTE network can exchange encrypted messages. With LTE network we mean MME, as it corresponds to the actual destination of all the messages sent by UE with NAS protocol, which is the protocol for control plane communications.

Please notice that NAS security is not enough. In fact, this allows to protect just the communication between UE and MME. In a radio message from the eNB to the UE, an eavesdropper can get all the details of the communication, except for the field that carries the information from the MME. In order to protect the remaining fields as well, an additional layer of security is needed, which is called AS security. It allows to grant secure communications over a wireless link, like the one between UE and eNB.

Further details about the security procedures and the state variables involved in them are available in [41, 22, 50, 51].

3.1.4 Session information

Here we list the information related to the data channel between the UE and the Internet.

- APN: identifier for the Packet Data Network (PDN), stored in the UE;
- Default APN: identifier for a PDN, used if UE does not provide a valid APN;
- APN In Use: identifier for the APN used eventually (may be equal to APN or to Default APN);
- P-GW ID: identifier for the P-GW;
- P-GW IP: the IP address of the P-GW;
- EPS Bearer ID: identifier for the whole data plane channel between UE, eNB, S-GW, and P-GW;
- DRB ID: identifier for the first part of the data plane channel, between UE and eNB. It comprises both uplink and downlink identifiers;

- E-RAB ID: identifier for the first two parts of the data plane channel, considered as a single channel (UE - eNB - S-GW). Please note that E-RAB ID and EPS Bearer ID correspond to the same value;
- S1 TEID UL: identifier for the uplink of the second part of the data plane channel (eNB - S-GW);
- S1 TEID DL: identifier for the downlink of the second part of the data plane channel (eNB - S-GW);
- S5 TEID UL: identifier for the uplink of the third part of the data plane channel (S-GW - P-GW);
- S5 TEID DL: identifier for the downlink of the third part of the data plane channel (S-GW - P-GW);
- UE-AMBR UL: maximum upload bandwidth for non-GBR bearers (PDN-independent value);
- UE-AMBR DL: maximum download bandwidth for non-GBR bearers (PDN-independent value);
- QCI_{HSS}: identifier for a certain set of quality treatments (identifies a certain quality of service). It is retrieved from the HSS;
- ARP_{HSS}: priority value associated to each bearer retrieved from the HSS;
- APN-AMBR UL_{HSS}: maximum upload bandwidth for non-GBR bearers (PDN-specific value) retrieved from the HSS;
- APN-AMBR DL_{HSS}: maximum download bandwidth for non-GBR bearers (PDN-specific value) retrieved from the HSS;
- QCI_{SPR}: identifier for a certain set of quality treatments (identifies a certain quality of service). It is retrieved from the SPR;
- ARP_{SPR}: priority value associated to each bearer retrieved from the SPR;
- APN-AMBR UL_{SPR}: maximum upload bandwidth for non-GBR bearers (PDN-specific value) retrieved from the SPR;
- APN-AMBR DL_{SPR}: maximum download bandwidth for non-GBR bearers (PDN-specific value) retrieved from the SPR;
- QCI_{PCC}: identifier for a certain set of quality treatments (identifies a certain quality of service). It is contained in the PCC rule;
- ARP_{PCC}: priority value associated to each bearer contained in the PCC rule;
- APN-AMBR UL_{PCC}: maximum upload bandwidth for non-GBR bearers (PDN-specific value) contained in the PCC rule;
- APN-AMBR DL_{PCC}: maximum download bandwidth for non-GBR bearers (PDN-specific value) contained in the PCC rule;
- TFT UL: filter that separates a single flow of upload packets into their bearers;

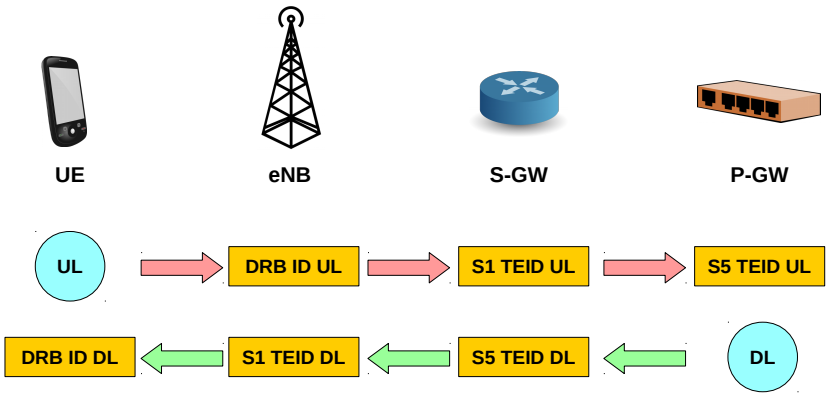


Figure 3.1: **The main tunnel identifiers of a default bearer.** *The red arrows represent the uplink channel, while the green arrows the downlink one. The blue circles represent the sources of uplink and downlink packets instead.*

- TFT DL: filter that separates a single flow of download packets into their bearers;
- SDF Filter: separates a flow of packets depending on the QoS policies to be applied.

There are some additional information that are used only during handover procedure. These are:

- UL/DL Count: indicates to the Target eNB (TeNB) from which packet it has to start the buffering;
- X2 TEID DL: tunnel endpoint of the TeNB on the X2 interface, used by the Source eNB (SeNB) to forward downlink packets;
- S1 eNB TEID DL: tunnel endpoint of the TeNB on the S1 interface, used by S-GW to forward downlink packets;
- S1 S-GW TEID UL: tunnel endpoint of the S-GW on the S1 interface, used by SeNB to forward downlink packets.

When a UE attaches to an LTE network (or transits from idle to active state), a data bridge between the UE and the Internet (which is called Packet Data Network in this context) has to be established. It is called default bearer, as it is used as default communication channel: as it is a best-effort communication channel, it belongs to the non-Guaranteed Bit Rate (non-GBR) bearers.

Figure 3.1 explains some of the identifiers we have previously listed. The communication channel between UE and eNB is a Data Radio Bearer (DRB), which is

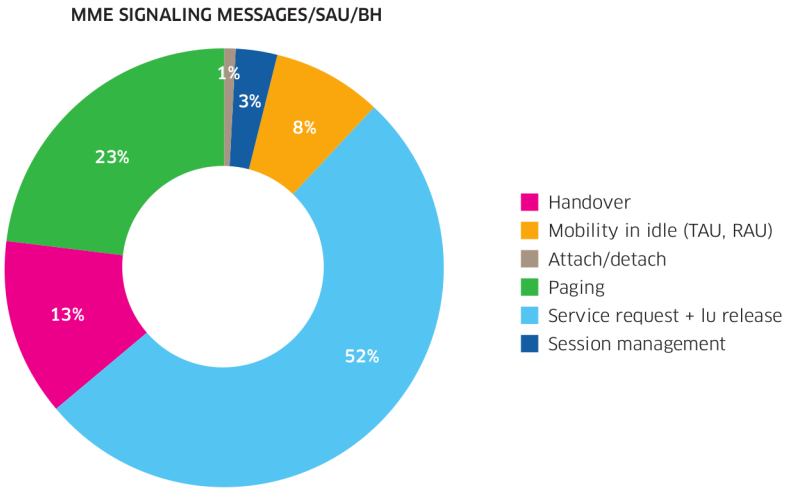


Figure 3.2: **Percentages of signals related with network events going through the MME.** *Most of the signals are related with transitions from Active to Idle state and vice-versa (Service request + Iu release and Paging). The chart is picked from [29].*

represented by its identifiers for upload and download. eNB and S-GW communicate through the S1 interface, and therefore they need some identifiers to multiplex the different user-specific communications on it: to this aim, each communication has a couple of Tunnel Endpoint IDentifier (TEID), one for upload and one for download. Similarly, S-GW and P-GW use the same technique to multiplex the different communications over the same S5 link.

Please notice that each arrow in the figure identifies handler and user of each variable. The tip of the arrow points to the handler of the variable, so the entity which is in charge of creating and distributing it, together with any related update. Instead, the tail of the arrows indicates the user of the variable. As an example, the UE needs to know the DRB UL ID in order to send uplink packets to the eNB, which is the actual handler of the considered variable.

Further details about the GPRS Tunnelling Protocol used for the data bearers in LTE networks are available in the technical specification [15].

3.2 The Network Events

An LTE network is an intrinsically reactive structure, in the sense that its operations are triggered by events that happen in the network: in this sense, the behaviour of the network is essentially user-driven. Here, we distinguish between network events and the corresponding procedures triggered. A network event is a situation created by the users of the network, as a UE exiting from the coverage area of a base station and entering in the one of another base station. A procedure is the sequence of steps carried out by the network in response to the event: in this case, the handover procedure is triggered. Please notice that in some circumstances we use the name of a procedure (possibly followed by the word 'event') to indicate the corresponding network event.

As the network has to handle a huge variety of events, we would like to focus on

the ones that mostly influence the impact of the signaling storm phenomenon. The chart in Figure 3.2 shows how the signals reaching and departing from the MME are subdivided among the different network events. We are considering the MME as it is the entity that is most affected by the signaling storm phenomenon. We can see how the transition from active to idle state and vice-versa represent the greatest source of signals for the MME. In fact, service requests is another way of calling transitions from idle to active state, as well as paging. Instead, the Iu releases can be mapped to the opposite kind of transitions.

The transitions imply 75% of the signals reaching the MME: this confirms our preliminary insights about the sources of the signaling storm phenomenon in Section 2. Moreover, we can see that handover operations entail another significant amount of signals (13%): this is also confirmed in the report from Nokia [84]. The impact of the attach procedure is very limited instead (1%), but it could become an issue in scenarios in which its procedure becomes the most frequent one, as huge sensor networks. Therefore, we have decided to focus our analysis on the following procedures:

- Initial attach;
- Transition from Active to Idle state;
- Transition from Idle to Active state (triggered by the UE);
- Transition from Idle to Active state (triggered by the network);
- Handover performed through the X2 interface (X2 handover);
- Handover performed without the X2 interface (S1 handover).

As in the previous section, we will avoid low-level details of the procedures, focusing on the main signals that are exchanged among the entities of the current LTE architecture. A general understanding of the network events and their related procedures is available at [42]. Moreover, additional details about the state variables carried by the signals going through the S1-MME interface are available in the technical specification [21]. Similarly, the X2 interface and its related signals are deeply explained in [23]. Details on the NAS control protocol (widely used during the considered procedures) and the signals that are exchanged using it can be found in [19].

Before going through the different procedures, we make a clarification about the transitions we are considering. Please notice that they are not related with discontinuous reception techniques (DRX), even if the aims are similar [27]. The transitions we are considering are related with the state of the UE, which can be active, meaning that the default connection with the LTE network is set up, and idle, in which the connection is (temporarily) released and most of the state variables are discarded both by UE and eNB. The UE transits to idle state after the expiration of an inactivity timeout [40], and it wakes up when uplink packets have to be sent to the Internet or when downlink packets are available for it. It can be considered a system-level technique. Instead, discontinuous reception allows to monitor the radio channel through an algorithm that switches on and off the radio interface, instead of keeping it always on. It is therefore a radio-level technique. Both transitions and DRX techniques aim to save battery power, but DRX takes place both during idle and active states [83, 66].

3.2.1 Initial Attach

Here we describes the different steps in the initial attach procedure. In particular, the case we take into account is the "very" initial attach, in the sense that no information about the UE (and about its subscriber) is in the network, except for the subscriber's information that are stored in the two databases (HSS and SPR). The detailed explanation of the procedure can be found in [43] and [53]. The procedure is graphically explained in [92], and the details about the information carried in each signal are available in Section A.1.

We can subdivide the initial attach procedure in 5 steps:

- UE ID acquisition;
- Authentication;
- NAS security setup;
- Location update;
- EPS session establishment.

3.2.1.1 UE ID acquisition

Our initial scenario consider a UE which has already selected the eNB to start the attach procedure. The first aim of a UE is to establish a control link with the LTE network, since a control link allows to exchange configuration parameters for creating data links later. No control intelligence is in the eNBs, so the control link must be established between the UE and a control entity of the LTE network, as the MME for example. Since no direct link exists between UEs and MME, the UE sets up a connection with the eNB first, then the eNB establishes a connection with the MME. After that, a message channel is established from the UE to the MME and vice-versa.

Therefore, the first step is the creation of a connection between the UE and the selected eNB. The first signals that UE and eNB exchange regulate the access to the wireless channel. Even if they are low-level messages, we highlight them, as they carry some meaningful information.

- Step 1: Random Access Preamble (UE $\rightarrow$ eNB);
- Step 2: Random Access Response (UE $\leftarrow$ eNB).

eNBs regulate the access to the wireless channel through two possible approaches: contention-based and contention-free [119, 36]. Here we assume a contention-based access, in which the 'scheduled transmission' message matches our Step 3, and no 'contention resolution' messages are needed. Further details about the procedures for the regulation of the accesses to the wireless channel are available in [24, 25].

After the UE is granted the access to the shared media, it can establish a communication channel with the eNB. Since we are dealing with control plane, we use a control plane protocol, which is the Radio Resource Control (RRC) protocol in this case. To set up an RRC connection, three steps are required:

- Step 3: RRC Connection Request (UE $\rightarrow$ eNB);
- Step 4: RRC Connection Setup (UE $\leftarrow$ eNB);
- Step 5: RRC Connection Setup Complete (UE $\rightarrow$ eNB).

Now that a channel between UE and eNB is available, we need another channel between eNB and MME. Since we are dealing with control plane, the protocol used between eNB and MME is S1. To set up a S1 connection, a single step is needed.

- Step 6: Initial UE Message (eNB → MME).

Please notice that the fifth message (RRC Connection Setup Complete) has the IMSI of the subscriber as embedded in one of its fields. This is embedded in turn into the Initial UE Message: in this way, the MME receives the IMSI of the subscriber and therefore the UE ID acquisition phase is completed.

3.2.1.2 Authentication

The authentication phase has the aim of mutually authenticate subscriber and network. We can split the authentication phase in two sub-phases:

- Acquisition of authentication vectors;
- Mutual authentication.

3.2.1.2.1 Acquisition of authentication vectors

The authentication information of each subscriber of the network are stored in the HSS. Therefore, the MME has to pick those information from there in order to perform the authentication with the UE. Two simple messages are exchanged:

- Step 7: Authentication Information Request (MME → HSS);
- Step 8: Authentication Information Answer (MME ← HSS).

3.2.1.2.2 Mutual authentication

In this phase, the UE recognizes that the network is really the one it wanted to join, in the same way as the network recognizes that the UE is of a real subscriber. Since there is no direct link between MME and UE, we used the channel created previously for the communication between them. This means that each message exchanged has actually to pass through the eNB. Two messages are exchanged:

- Step 9: Authentication Request (eNB ← MME);
- Step 10: Authentication Request (UE ← eNB);
- Step 11: Authentication Response (UE → eNB);
- Step 12: Authentication Response (eNB → MME).

3.2.1.3 NAS security setup

NAS is the protocol used between UE and MME in order to exchange control plane messages. In order to ensure security in the communication between them, they exchange some security parameters. We just have two messages in this situation as well (which become four since eNB relays the messages):

- Step 13: Security Mode Command (eNB ← MME);
- Step 14: Security Mode Command (UE ← eNB);
- Step 15: Security Mode Complete (UE → eNB);
- Step 16: Security Mode Complete (eNB → MME).

3.2.1.4 Location update

The aim of this step is to store in the HSS the current position of the UE (actually just the MME to which the UE is registered) and to get from the HSS the QoS profile of the subscriber, such that the channels for user data can be set up accordingly. Just two messages are exchanged:

- Step 17: Update Location Request (MME → HSS);
- Step 18: Update Location Answer (MME ← HSS).

3.2.1.5 EPS session establishment

The aim of the last phase is to set up the data plane channel, such that data packets can flow from UE to the Internet and vice-versa. This kind of channel is called EPS bearer. The messages exchanged among the entities of the LTE network follows:

- Step 19: Create Session Request (MME → S-GW);
- Step 20: Create Session Request (S-GW → P-GW);
- Step 21: EPS Session Establishment Notification (P-GW → PCRF);
- Step 22: Profile Request (PCRF → SPR);
- Step 23: Profile Response (PCRF ← SPR);
- Step 24: EPS Session Establishment Ack (P-GW ← PCRF);
- Step 25: Create Session Response (S-GW ← P-GW);
- Step 26: Create Session Response (MME ← S-GW);
- Step 27: Initial Context Setup Request (eNB ← MME);
- Step 28: Security Mode Command (UE ← eNB);
- Step 29: Security Mode Complete (UE → eNB);
- Step 30: RRC Connection Reconfiguration (UE ← eNB);
- Step 31: Initial Context Setup Response (eNB → MME);
- Step 32: Attach Complete (UE → eNB);
- Step 33: Attach Complete (eNB → MME);
- Step 34: Modify Bearer Request (MME → S-GW);
- Step 35: Modify Bearer Response (MME ← S-GW).

The data channel is composed by three parts: the channel from UE to eNB (DRB bearer), the channel from eNB to S-GW (S1 bearer) and the channel from S-GW to P-GW (S5 bearer).

In Steps 19-20, S-GW forwards the request of the MME to the P-GW. P-GW then asks to the PCRF the policy rules it has to apply on the data channel (Step 21). PCRF retrieves from the SPR the subscriber details (Steps 22-23) in order to create the policy rules, and then sends them to the P-GW (Step 24). The P-GW creates the

data channel with S-GW, which notifies the MME in turn (Steps 25-26). Now the last part of the whole channel (S5 bearer) is ready.

In Step 27, the parameters for establishing the first part of the channel are sent to the eNB. The eNB first secures the channel with UE (Steps 28-29) and then it sends to the UE the parameters (QoS parameters) for creating the data channel (Step 30). After that, the first and the last parts of the whole channel are ready. The eNB notifies the MME that no issues have been encountered (Step 31).

Finally, the part in the middle of the channel has to be set up. The UE communicates to the MME that it has ended the attach procedure (Steps 32-33). Therefore, the MME sends the parameters for the creation of the middle part of the channel to the S-GW (Step 34), and the S-GW answers back once completed (Step 35).

Figure 3.3 shows the described steps.

3.2.2 Active_to_Idle Transition

This section aims to explain the messages that are exchanged when the UE goes from Active state to Idle state. As we have seen from the initial attach procedure, we have essentially two channels to deal with:

- the control channel (i.e. ECM connection), composed by an RRC connection (UE - eNB) and an S1 connection (eNB - MME);
- the data channel (i.e. EPS bearer), composed by a DRB bearer (UE - eNB), an S1 connection (eNB - S-GW) and an S5 connection (S-GW - P-GW).

When Idle state is reached, all the pieces of channels are released, except for the S5 connection (S-GW - P-GW) of the data channel[1]. Please notice that the control channels between MME, S-GW and P-GW are always-on and independent from the UEs, meaning that they can always exchange control plane messages regardless the presence of UEs.

The messages that are exchanged in this phase follow. The details of the procedure can be found in [54]. The procedure is graphically explained in [89], and the details about the information carried in each signal are available in Section A.2.

- Step 1: UE Context Release Request (eNB $\rightarrow$ MME);
- Step 2: Release Access Bearers Request (MME $\rightarrow$ S-GW);
- Step 3: Release Access Bearers Response (MME $\leftarrow$ S-GW);
- Step 4: UE Context Release Command (eNB $\leftarrow$ MME);
- Step 5: RRC Connection Release (UE $\leftarrow$ eNB);
- Step 6: UE Context Release Complete (eNB $\rightarrow$ MME).

User inactivity can be detected by the eNB, or the UE can explicitly send a message to the eNB its will to go in Idle state: in the latter case, we have a additional "Step 0" (UE $\rightarrow$ eNB) in which the UE informs the eNB. Then the eNB asks the MME to release the resources associated with the UE (Step 1). MME contacts the S-GW in order to instruct it to do the same (Step 2), then the latter confirms the operation

[1] Actually, the user plane S1 connection is not completely torn down, since the uplink is kept active. This allows the uplink packets to reach the S-GW without setting up a channel again.

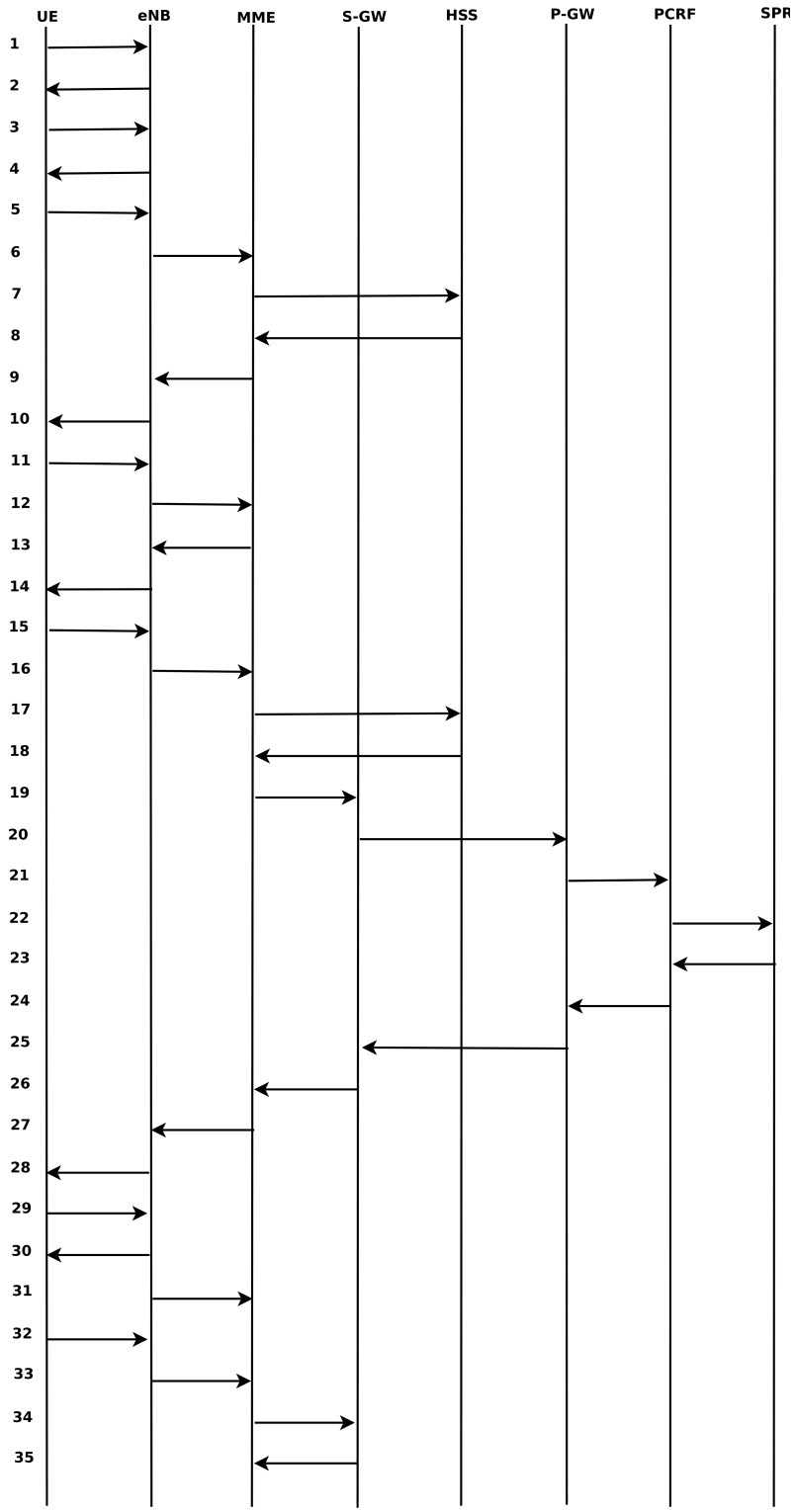


Figure 3.3: **Sequence diagram of the initial attach procedure.** *The procedure is particularly heavy for the LTE architecture, as it requires 35 signals in total.*

(Step 3). No messages are sent to the P-GW since the S5 connection must be kept active. Please notice that the S-GW will buffer any incoming downlink packet for the UE till the re-activation of the connection. The MME answers back to the eNB, since all the other components of the LTE network have been properly configured (Step 4). Then the eNB instructs the UE to release all the connections with it (Step 5) and it confirms the completion of the operation to the MME (Step 6).

Figure 3.4 shows the described steps.

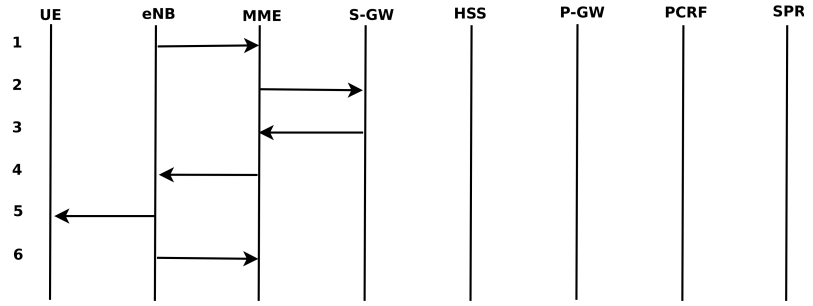


Figure 3.4: **Sequence diagram of the Active_to_Idle transition.** *After the procedure, the first two parts of the data bearer (DRB and S1) are released, while the S5 part is still active.*

3.2.3 Idle_to_Active Transition

A UE transits from Idle state to Active state when one of the following two situations happens:

- the UE has uplink packets to deliver to the network (UE-triggered transition);
- the network has downlink packets to deliver to the UE (network-triggered transition).

We analyze the two situations separately. The details of the procedures can be found in [55].

3.2.3.1 UE Triggered

We first describe the situation in which the UE has uplink packets to deliver to the network. Therefore, the UE needs to transit from Idle state to Active state, meaning that the resources for it have to be re-allocated.

The procedure is graphically explained in [91], and the details about the information carried in each signal are available in Section A.3. The sequence of the messages that are exchanged follows.

- Step 1: Random Access Preamble (UE $\rightarrow$ eNB);
- Step 2: Random Access Response (UE $\leftarrow$ eNB);
- Step 3: RRC Connection Request (UE $\rightarrow$ eNB);
- Step 4: RRC Connection Setup (UE $\leftarrow$ eNB);

- Step 5: RRC Connection Setup Complete (UE → eNB);
- Step 6: Initial UE Message (eNB → MME);
- Step 7: Initial Context Setup Request (eNB ← MME);
- Step 8: AS Security Mode Command (UE ← eNB);
- Step 9: AS Security Mode Complete (UE → eNB);
- Step 10: RRC Connection Reconfiguration (UE ← eNB);
- Step 11: Initial Context Setup Response (eNB → MME);
- Step 12: Modify Bearer Request (MME → S-GW);
- Step 13: Modify Bearer Response (MME ← S-GW).

Please notice that the sequence of steps is just a lightened version of the Initial Attach Procedure. The only difference is that the UE context has not to be created from scratch, but the previously stored one can be used to avoid several steps instead. The first six steps are almost identical to the ones in the Initial Attach Procedure. Steps 1 and 2 regulate the access to the wireless channel, and Steps 3 and 4 allow to set up the RRC channel, which is the first part of the control channel. Step 5 is used to carry the actual service request (the need of uploading packets) to the MME, and then the service request flows through the eNB (Step 6). The service request has exactly the same treatment of the IMSI in the Initial Attach procedure: it is carried by both the RRC Connection Setup Complete and the Initial UE Message in turn. The UE context is still available in the MME, so it can be used in order to skip a lot of message exchanges. The UE context is given to the eNB (Step 7) such that the first part of the user plane channel can be set up. The eNB first secures the communication with the UE (Steps 8-9), then it creates the new DRB bearer (Step 10). The eNB acknowledges the operation to the MME (Step 11), which in turn takes care to create the middle part of the channel (the S1 bearer) together with the S-GW (Steps 12-13).

In order to present the simplest scenario, we have made some assumptions that reduce to the minimum the number of exchanged messages:

- the integrity check performed by the MME on the NAS security context passes (after Step 6), such that a new authentication with UE is not needed again;
- the cell has not changed from the last Active_to_Idle transition (nor the tracking area consequently), such that no modifications of the session are needed (after Step 12).

Figure 3.5 shows the described steps.

3.2.3.2 Network Triggered

When the network detects the presence of packets that have to be delivered to the UE, it has to inform the UE, such that the procedure described in the previous section can take place. Therefore the only difference with the previous procedure is in an additional initial part, called paging, in which the network contacts the UE to inform it of the presence of downlink packets.

The procedure is graphically explained in [90], and the details about the information carried in each signal are available in Section A.4. The sequence of the messages that are exchanged follows.

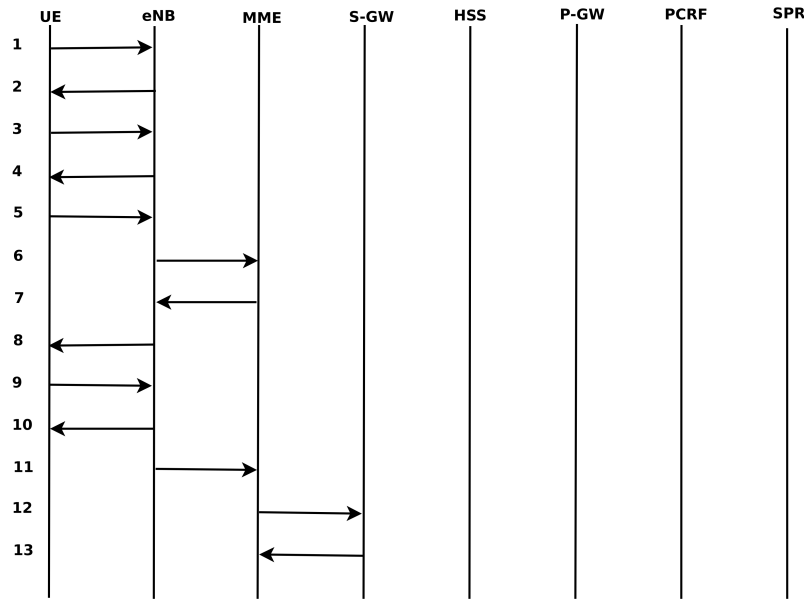


Figure 3.5: **Sequence diagram of the Idle_to_Active transition (UE triggered).** *The procedure consists in a light version of the initial attach procedure, as many steps are skipped thanks to the UE context stored in the MME.*

- Step 1: Downlink Data Notification (MME ← S-GW);
- Step 2: Downlink Data Notification Ack (MME → S-GW);
- Step 3: Paging (eNB ← MME);
- Step 4: Paging (UE ← eNB);
- Step 5: Random Access Preamble (UE → eNB);
- Step 6: Random Access Response (UE ← eNB);
- Step 7: RRC Connection Request (UE → eNB);
- Step 8: RRC Connection Setup (UE ← eNB);
- Step 9: RRC Connection Setup Complete (UE → eNB);
- Step 10: Initial UE Message (eNB → MME);
- Step 11: Initial Context Setup Request (eNB ← MME);
- Step 12: AS Security Mode Command (UE ← eNB);
- Step 13: AS Security Mode Complete (UE → eNB);
- Step 14: RRC Connection Reconfiguration (UE ← eNB);
- Step 15: Initial Context Setup Response (eNB → MME);
- Step 16: Modify Bearer Request (MME → S-GW);

- Step 17: Modify Bearer Response (MME ← S-GW).

Incoming downlink packets come from the P-GW to the S-GW, which is not able to deliver them as no resources are allocated. Therefore, the S-GW starts buffering the packets, and notifies the UE to start the Idle_to_Active transition at the same time. First, the S-GW informs the MME about the presence of new downlink packets (Step 1). The MME acknowledges the message (Step 2) and then it starts the Paging procedure (*i.e.* informing the UE about the incoming packets). Essentially, the MME has to reach the UE, but it only knows that the UE is in one of the tracking areas under its control (otherwise the UE would be under the handling of another MME). Therefore, the MME sends a Paging message to all the eNBs of the tracking areas handled (Step 3), and the eNBs broadcast in turn the Paging message in order to reach the UE (Step 4). At this point, the target UE is informed about the incoming downlink packets[2], so it starts all the steps of the traditional procedure, as described in the previous section. Please notice that we are making the same two "simplifying" assumptions of the previous case.

Figure 3.6 shows the described steps.

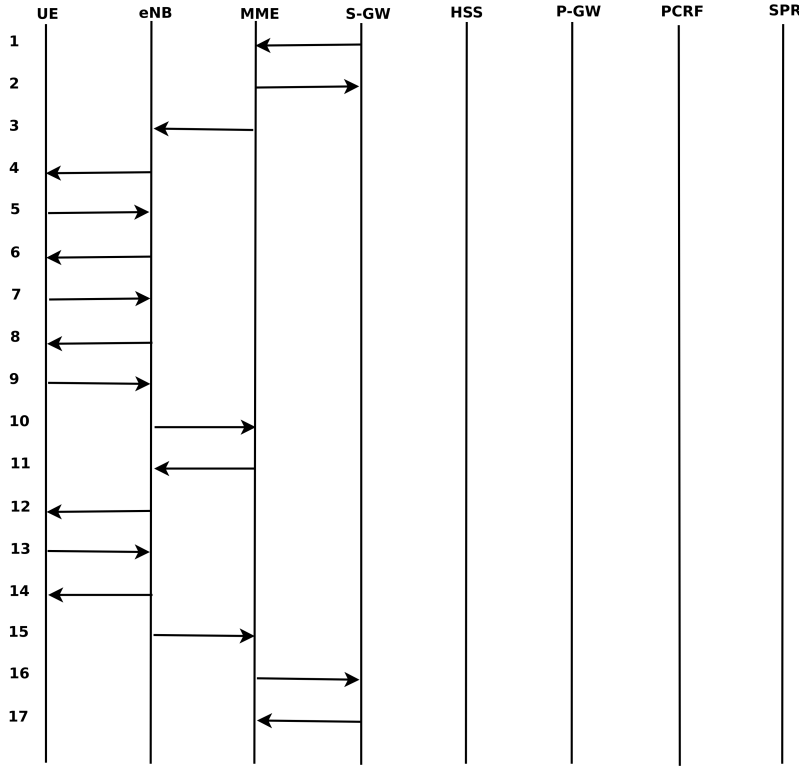


Figure 3.6: **Sequence diagram of the Idle_to_Active transition (network triggered).** *The four initial signals are used to notify the UE of the incoming downlink packets, while the rest of the procedure equals to the UE-triggered case.*

[2]The UE in Idle state periodically monitors the radio channel for paging messages.

3.2.4 Handover

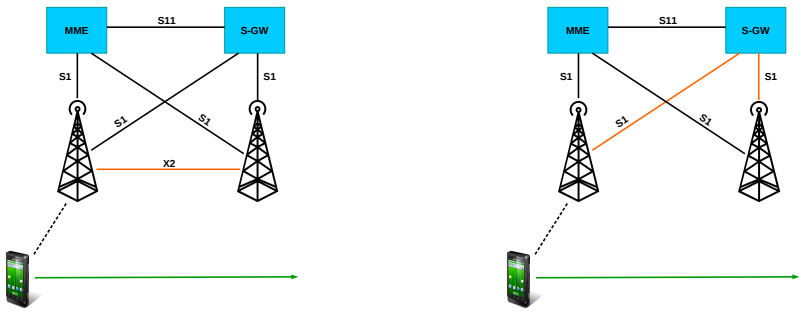
In this section, we discuss the steps performed by the LTE architecture during an handover. In order to avoid too specific details, we will focus on the most simple case of handover. The handover we examine is between two eNBs that share the same MME and the same S-GW: this means that the two eNBs belong to the tracking areas that are under control of the MME, therefore no Tracking Area Update (TAU) is needed. We will however distinguish between two cases:

- an X2 interface directly links the two involved eNBs (Figure 3.7a);
- the two involved eNBs have no X2 interface linking them (Figure 3.7b).

The first case is the simplest scenario, and it is called X2 handover, while the second one requires an indirect link between the eNBs through the S-GW, and it is therefore called S1 handover[3]. Regardless the kind of handover, the procedure is mainly divided in three parts:

- Preparation;
- Execution;
- Completion.

The following section discuss the two kinds of handover, exploring the steps performed in each of the three phases. The detailed description of the procedure is available in [56].



(a) X2 Handover

(b) S1 Handover

Figure 3.7: **The two typologies of handover.** *The orange lines highlight the path that is actually used to let the two eNBs communicating.*

3.2.4.1 X2 Handover

The handover is triggered once the eNB to which a UE is attached receives a Measurement Report from the UE. In order to trigger the handover, the signal from the UE must report a particular event. In our scenario, the signal strength of a neighbour cell (TeNB, the target base station) has become stronger that the signal strength of the current cell (SeNB, the source base station). Therefore, the very first step is:

[3]We are not considering the possibility that the X2 interface doesn't support handover, or the case in which the preparation of the handover was unsuccessful. In these situations, an S1 handover is executed instead.

- Step 1: Measurement Report (UE → SeNB).

The subsequent steps are reported and explained separately for each sub-phase. The details of the procedures can be found in [57]. The procedure is graphically explained in [95], and the details about the information carried in each signal are available in Section A.5.

3.2.4.1.1 Preparation

The steps are first presented and then explained, as usual:

- Step 2: Handover Request (SeNB → TeNB);
- Step 3: Handover Request Ack (SeNB ← TeNB).

The Measurement Report triggers the handover, and since an X2 interface is connecting the two eNBs, they can communicate directly. Therefore, the SeNB sends an Handover Request to TeNB (Step 2) in which the UE context is included. The TeNB tests if it will be able to allocate the resources requested by the UE (the requirements are written in the UE context). After that, the TeNB establishes a S1 data plane channel with the S-GW, and then it sends an Handover Request Ack to the SeNB (Step 3). The ack carries a temporary ID that the UE uses once attached to the new cell. Moreover, the ack has the information needed to create an X2 data plane bearer between the two eNBs, which is therefore created by the SeNB at the reception of the ack. The channel will be used later to forward to the TeNB the downlink packets that are still going from the S-GW to the SeNB.

3.2.4.1.2 Execution

The following steps are performed during the execution phase:

- Step 4: Handover Command (UE ← SeNB);
- Step 5: Sequence Number Status Transfer (SeNB → TeNB);
- Step 6: Handover Confirm (UE → TeNB).

Once received the acknowledgement from the TeNB, the SeNB instructs the UE to perform the handover. An Handover Command is sent to the UE embodied in an RRC Connection Reconfiguration message (Step 4). While the UE changes base station, the SeNB sends a Sequence Number Status Transfer to the TeNB (Step 5): in this way, the TeNB knows from which point of the packet flow it should start buffering the downlink packets for the UE. From now on, the SeNB will forward all the packets for the UE through the X2 data plane channel to the TeNB, which buffers them in order to give them later to the UE. Once the handover is completed, the UE sends a Handover Confirm to the TeNB through a RRC Connection Reconfiguration Complete message (Step 6). After that, a DRB bearer is established between the UE and the TeNB, such that data plane packets can flow between them. Please note that there is no need of creating a RRC control channel between the UE and the TeNB, because it has been automatically built with the information received form the SeNB (essentially, the control channel has been redirected from SeNB to TeNB). Same holds for the S1 control channel (from SeNB-MME to TeNB-MME).

3.2.4.1.3 Completion

The list of the final steps follows:

- Step 7: Path Switch Request (TeNB → MME);
- Step 8: Modify Bearer Request (MME → S-GW);
- Step 9: Modify Bearer Request (S-GW → P-GW);
- Step 10: EPS Session Modification Notification (P-GW → PCRF);
- Step 11: EPS Session Modification Ack (P-GW ← PCRF);
- Step 12: Modify Bearer Response (S-GW ← P-GW);
- Step 13: Modify Bearer Response (MME ← S-GW);
- Step 14: Path Switch Request Ack (TeNB ← MME);
- Step 15: UE Context Release (SeNB ← TeNB).

Considering the two channels needed (the control plane one and the data plane one), now we just need the piece of data plane channel which connects the new eNB (TeNB) to the S-GW. For this reason, a Path Switch Request is sent from the TeNB to the MME (Step 7), to instruct the S-GW about the new data plane channel. The MME sends in turn the request to the S-GW through a Modify Bearer Request (Step 8). The S-GW reacts redirecting the S1 data plane channel towards TeNB: now the channels are completed. In some cases, the PCRF has to be notified every time the UE changes cell, in order to take into account this information when policy rules for that UE are created. Therefore, the S-GW forwards the same Modify Bearer Request to the P-GW (Step 9), which then transmits the information to the PCRF through the EPS Session Modification Notification (Steps 10). Then a sequence of acknowledgements follows (Steps 11-14): the main aim is to reach the source of the first message (the TeNB) in order to communicate that no issues have been experienced. The handover is then completed. Therefore, the SeNB is instructed by the TeNB to release the resources allocated for the UE, which has just moved (Step 15).

Figure 3.8 shows the described steps.

3.2.4.2 S1 Handover

The S1 handover is a procedure triggered once the conditions for the X2 handover are met, but at the same time an X2 interface connecting the two interested base stations (SeNB and TeNB) is missing. In the previous case of handover, a channel between the two base stations was created in order to allow the packets to be forwarded from the SeNB to the TeNB during the handover procedure. Since this kind of channel is required, in S1 handover the channel is created through the S-GW. Similarly to the X2 handover, the procedure is triggered after the Measurement Report of the UE:

- Step 1: Measurement Report (UE → SeNB).

The subsequent steps are reported and explained separately for each sub-phase. The details of the procedures can be found in [58]. The procedure is graphically explained in [93], and the details about the information carried in each signal are available in Section A.6.

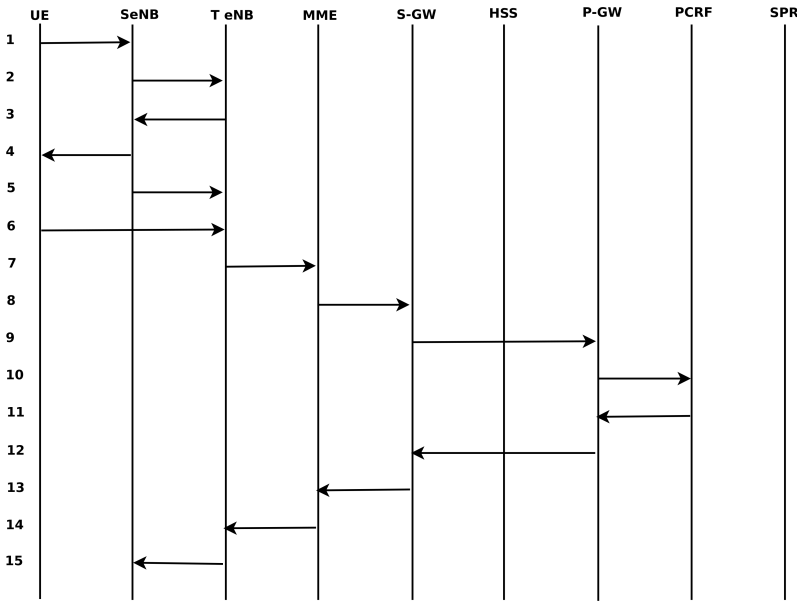


Figure 3.8: **Sequence diagram of the X2 handover.** *Thanks to the X2 interface linking the two base stations, most of the burden related to the handover is handled directly between them.*

3.2.4.2.1 Preparation

The steps of the preparation phase are:

- Step 2: Handover Required (SeNB → MME);
- Step 3: Handover Request (TeNB ← MME);
- Step 4: Handover Request Ack (TeNB → MME);
- Step 5: Create Indirect Data Forwarding Tunnel Request (MME → S-GW);
- Step 6: Create Indirect Data Forwarding Tunnel Response (MME ← S-GW);
- Step 7: Handover Command (SeNB ← MME).

Since the SeNB cannot communicate directly with the TeNB, it sends an Handover Required message to the MME (Step 1). The MME then sends an Handover Request message to the TeNB (Step 2) asking if it is able to grant the resources for the UE. In this step, the UE context is transmitted to the TeNB as well. The TeNB answers back to the MME with a Handover Request Ack (Step 4) after the resources allocation. Then, the MME instructs the S-GW to create a channel with the TeNB: this is the first part of the indirect link between SeNB and TeNB through S-GW. These instructions are embodied in the message Create Indirect Data Forwarding Tunnel Request (Step 5). The S-GW answers back to the MME, providing also the information needed for the setup of the second part of the indirect link (between SeNB and S-GW) through the Create Indirect Data Forwarding Tunnel Response (Step 6). Finally, when MME receives the message, it forwards these information to the SeNB through a Handover Command message (Step 7), such that the indirect link can be completed. The message signals that the handover can start.

3.2.4.2.2 Execution

The list of the steps performed during the execution phase follows:

- Step 8: Handover Command (UE ← SeNB);
- Step 9: eNB Status Transfer (SeNB → MME);
- Step 10: MME Status Transfer (TeNB ← MME);
- Step 11: Handover Confirm (UE → TeNB).

The steps resemble the ones performed in X2 handover. The SeNB has to inform the TeNB from which packet it should start the buffering for the UE, similarly to the previous handover case. The Handover Command (Step 1) instructs the UE to actually start the handover, detaching from the current cell and attaching to the new one. The message is embodied into a RRC Connection Reconfiguration message. Therefore the SeNB sends the 'packets status' to the TeNB through the MME (Steps 9-10), as it is a control plane information and therefore we cannot use the indirect link (through S-GW) previously created. From now on, the SeNB can forwards the packets for the UE to TeNB, which buffers them. Finally, the UE completes the handover procedure sending a Handover Confirm (Step 11) to the TeNB through a RRC Connection Reconfiguration Complete. Now the DRB bearer between UE and TeNB is established, so the TeNB can start forwarding the downlink buffered packets to the UE.

3.2.4.2.3 Completion

As before, we need to create the S1 bearer for data plane packets between TeNB and S-GW, and to release the resources still allocated for the UE in SeNB. The final steps of the S1 handover are:

- Step 12: Handover Notify (TeNB → MME);
- Step 13: Modify Bearer Request (MME → S-GW);
- Step 14: Modify Bearer Request (S-GW → P-GW);
- Step 15: EPS Session Modification Notification (P-GW → PCRF);
- Step 16: EPS Session Modification Ack (P-GW ← PCRF);
- Step 17: Modify Bearer Response (S-GW ← P-GW);
- Step 18: Modify Bearer Response (MME ← S-GW);
- Step 19: UE Context Release Command (SeNB ← MME);
- Step 20: UE Context Release Complete (SeNB → MME);
- Step 21: Delete Indirect Data Forwarding Tunnel Request (MME → S-GW);
- Step 22: Delete Indirect Data Forwarding Tunnel Response (MME ← S-GW).

The Handover Notify message from the TeNB to the MME (Step 12) has the same aim of the Path Switch Request of the X2 handover: it instructs the S-GW to create the new data plane channel (between TeNB and S-GW). Therefore, the MME sends the message to the S-GW through the Modify Bearer Request (Step 13). Steps 13-18

correspond exactly to Steps 8-13 of the completion phase of the X2 handover. The final steps take care about the release of the resources: first the SeNB is instructed to release the resources for the UE (Steps 19-20), and then S-GW is instructed to release the channels of the indirect link previously used between the two base stations (Steps 21-22).

Figure 3.9 shows the described steps.

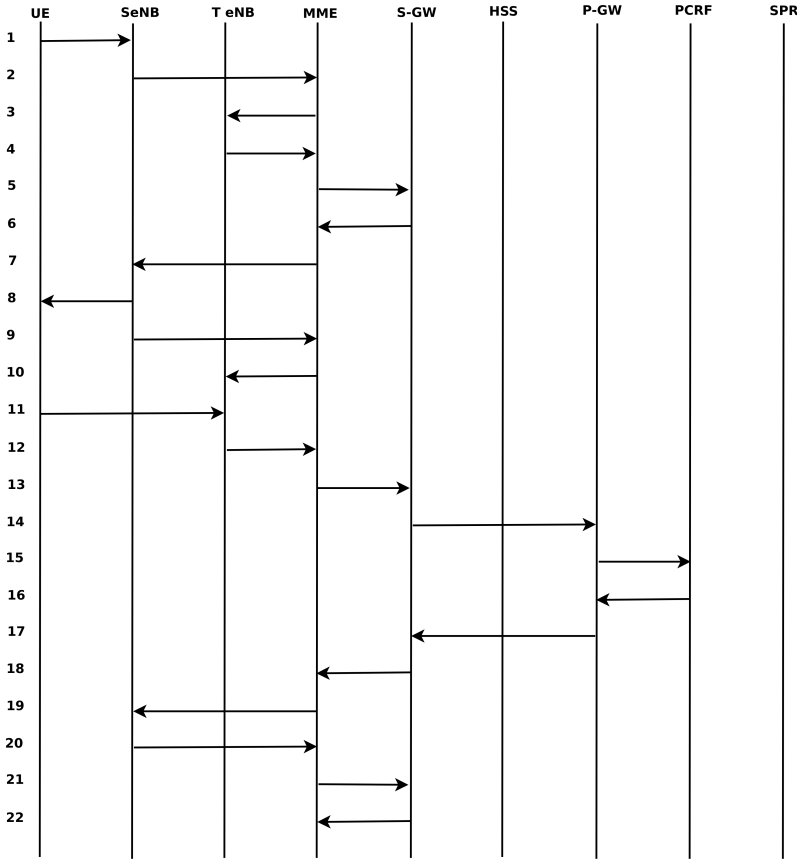


Figure 3.9: **Sequence diagram of the S1 handover.** *The lack of the direct communication channel between the eNBs leads to a significant increase in the number of signals needed.*

3.3 Signaling Load Analysis

In this section, we provide the equations that allow to compute the total time requested by the current LTE architecture to handle each one of the considered events. In particular, we associate a time cost to each interface, and we use the frequency with which it is used in every procedure in order to get total time requested, both by each procedure and by each interface.

3.3.1 Interfaces

The aim of this section is to present the variables which will be used during the analysis of the time requested by the procedures. The list of variables follows:

- $c1$ corresponds to the cost associated to interface LTE-Uu, between UE and eNB;
- $c2$ corresponds to the cost associated to interface S1-U, between eNB and S-GW;
- $c3$ corresponds to the cost associated to interface S5, between S-GW and P-GW;
- $c4$ corresponds to the cost associated to interface S1-MME, between eNB and MME;
- $c5$ corresponds to the cost associated to interface S11, between S-GW and MME;
- $c6$ corresponds to the cost associated to interface Gx, between P-GW and PCRF;
- $c7$ corresponds to the cost associated to interface S6a, between MME and HSS;
- $c8$ corresponds to the cost associated to interface Sp, between PCRF and SPR;
- $c9$ corresponds to the cost associated to the interface X2, between two eNBs (if the eNBs are linked).

3.3.2 Initial Attach

The list of messages exchanged during Initial Attach follows:

- 13 messages exchanged over LTE-Uu interface (Steps 1-2-3-4-5-10-11-14-15-28-29-30-32);
- 0 messages exchanged over S1-U interface;
- 2 messages exchanged over S5 interface (Steps 20-25);
- 8 messages exchanged over S1-MME interface (Steps 6-9-12-13-16-27-31-33);
- 4 messages exchanged over S11 interface (Steps 19-26-34-35);
- 2 messages exchanged over Gx interface (Steps 21-24);
- 4 messages exchanged over S6a interface (Steps 7-8-17-18);
- 2 messages exchanged over Sp interface (Steps 22-23);
- 0 messages exchanged over X2 interface.

The total amount of messages required by the procedure is 35. The total amount of time requested by the message exchanging in the procedure can be expressed as:

$$tot\_time = 13 * c1 + 2 * c3 + 8 * c4 + 4 * c5 + 2 * c6 + 4 * c7 + 2 * c8$$

3.3.3 Active_to_Idle Transition

The list of messages exchanged during Active_to_Idle Transition follows:

- 1 message exchanged over LTE-Uu interface (Step 5);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 3 messages exchanged over S1-MME interface (Steps 1-4-6);
- 2 messages exchanged over S11 interface (Steps 2-3);
- 0 messages exchanged over Gx interface;
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over Sp interface;
- 0 messages exchanged over X2 interface.

The total amount of messages required by the procedure is 6. The total amount of time requested by the message exchanging in the procedure can be expressed as:

$$tot\_time = c1 + 3 * c4 + 2 * c5$$

3.3.4 Idle_to_Active Transition

In this section, we provide the equations related to the Idle_to_Active transition. First, we perform the analysis on the UE-triggered transition, and then we do the same for the network-triggered one.

3.3.4.1 UE Triggered

The list of messages exchanged during Idle_to_Active Transition - UE Triggered follows:

- 8 messages exchanged over LTE-Uu interface (Steps 1-2-3-4-5-8-9-10);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 3 messages exchanged over S1-MME interface (Steps 6-7-11);
- 2 messages exchanged over S11 interface (Steps 12-13);
- 0 messages exchanged over Gx interface;
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over Sp interface;
- 0 messages exchanged over X2 interface.

The total amount of messages required by the procedure is 13. The total amount of time requested by the message exchanging in the procedure can be expressed as:

$$tot\_time = 8 * c1 + 3 * c4 + 2 * c5$$

3.3.4.2 Network Triggered

The list of messages exchanged during Idle_to_Active Transition - Network Triggered follows:

- 9 messages exchanged over LTE-Uu interface (Steps 4-5-6-7-8-9-12-13-14);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 4 messages exchanged over S1-MME interface (Steps 3-10-11-15);
- 4 messages exchanged over S11 interface (Steps 1-2-16-17);
- 0 messages exchanged over Gx interface;
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over Sp interface;
- 0 messages exchanged over X2 interface.

The total amount of messages required by the procedure is 17. The total amount of time requested by the message exchanging in the procedure can be expressed as:

$$tot\_time = 9 * c1 + 4 * c4 + 4 * c5$$

3.3.5 Handover

Similarly to the Idle_to_Active transition, we have to analyze both variants of the procedure. Therefore, we first provide the equation related to the X2 handover, and then we provide the equation associated to the S1 handover.

3.3.5.1 X2 Handover

The list of messages exchanged during X2 Handover follows:

- 3 messages exchanged over LTE-Uu interface (Steps 1-4-6);
- 0 messages exchanged over S1-U interface;
- 2 messages exchanged over S5 interface (Steps 9-12);
- 2 messages exchanged over S1-MME interface (Steps 7-14);
- 2 messages exchanged over S11 interface (Steps 8-13);
- 2 messages exchanged over Gx interface (Steps 10-11);
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over Sp interface;
- 4 messages exchanged over X2 interface (Steps 2-3-5-15).

The total amount of messages required by the procedure is 15. The total amount of time requested by the message exchanging in the procedure can be expressed as:

$$tot\_time = 3 * c1 + 2 * c3 + 2 * c4 + 2 * c5 + 2 * c6 + 4 * c9$$

3.3.5.2 S1 Handover

The list of messages exchanged during S1 Handover follows:

- 3 messages exchanged over LTE-Uu interface (Steps 1-8-11);
- 0 messages exchanged over S1-U interface;
- 2 messages exchanged over S5 interface (Steps 14-17);
- 9 messages exchanged over S1-MME interface (Steps 2-3-4-7-9-10-12-19-20);
- 6 messages exchanged over S11 interface (Steps 5-6-13-18-21-22);
- 2 messages exchanged over Gx interface (Steps 15-16);
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over Sp interface;
- 0 messages exchanged over X2 interface.

The total amount of messages required by the procedure is 22. The total amount of time requested by the message exchanging in the procedure can be expressed as:

$$tot\_time = 3 * c1 + 2 * c3 + 9 * c4 + 6 * c5 + 2 * c6$$

3.3.6 Summary

Table 3.1 summarizes the results from the previous sections.

	IA	**AtI**	**ItA (UE)**	**ItA (Net)**	**X2H**	**S1H**	**Total**
LTE-Uu	13	1	8	9	3	3	37
S1-U	0	0	0	0	0	0	0
S5	2	0	0	0	2	2	6
S1-MME	8	3	3	4	2	9	29
S11	4	2	2	4	2	6	20
Gx	2	0	0	0	2	2	6
S6a	4	0	0	0	0	0	4
Sp	2	0	0	0	0	0	2
X2	0	0	0	0	4	0	4
Total	35	6	13	17	15	22	

Table 3.1: **Signaling load analysis of the current LTE architecture.** *The initial attach and the S1 handover represent the most demanding procedures, while LTE-Uu, S1-MME, and S11 corresponds to the mostly used interfaces.*

Table 3.1 shows that the most used interfaces are LTE-Uu, S1-MME and S11, while the most onerous events are the initial attach and the S1 handover. Since our refactoring process cannot modify the interface with the widespread LTE-enabled devices, we have to focus on the reduction of the signals on the other interfaces. Please notice that S1-MME interface links eNB and MME, and S11 interface links S-GW and MME. This suggests to put the functionalities of the MME either close to the RAN or close to the gateways internal to the mobile core to reduce the signaling load. Another

way of achieving the same goal is to shift the functionalities assigned to eNB and S-GW to a control entity, as the MME, such that the signals in charge of replicating the state variables needed by the tasks in the eNB and in the S-GW are no more needed. Our refactoring process considers both ways in different moments.

3.4 The Network Functions

In this section, we present which are the tasks that are carried out by the current LTE architecture when focusing on the network events presented previously. The idea behind is the following: from the convoluted LTE architecture, we abstract which are the functionalities performed, and we recognize which are the elements that are taking care of them. In particular, we consider the tasks as separated network functions: this is particularly useful during the subsequent refactoring process, as we are allowed to shift them among the components of the architecture leveraging the NFV paradigm. Indeed, we aim to gather functionalities working on the same information in the same entity, such that signals carrying those information are no more needed.

We label each task with an identifier, such that it is easier to assign the tasks to the entities during the refactoring process. Please notice that even if most of the tasks are carried out by a single device, some of them need the collaboration of many devices: the tasks executed by a single entity are labelled with the 'S' letter, while the tasks executed by multiple entities are labelled with the 'M' letter. The list of tasks follows.

- [S01] GUTI handling: creation and assignment of the temporary identifier to be used in place of IMSI;
- [S02] IP handling: creation and assignment to the UE of the IP address;
- [S03] C-RNTI handling: creation and assignment of the radio identifier (C-RNTI);
- [S04] MME S1AP ID handling: creation and assignment of the identifier related to the (control plane) tunnel between eNB and MME (MME part);
- [S05] eNB S1AP ID handling: creation and assignment of the identifier related to the (control plane) tunnel between eNB and MME (eNB part);
- [S06] TAI List / TAU Timer Value handling: creation and distribution of the information that regulate the tracking area updates;
- [S07] MME ID notification: the MME ID value is provided to the HSS;
- [S08] HSS authentication request: retrieval of the information needed for authentication between UE and the rest of the network from HSS;
- [S09] Security parameters generation: creation of the security parameters (valued used during authentication and security keys) using the information retrieved from the HSS;
- [S10] Choice of the NAS Security algorithm: choice of the algorithm used during NAS Security Setup;
- [S11] Choice of the AS Security algorithm: choice of the algorithm used during AS Security Setup;

- [S12] Choice of the APN to use: choice between the Default APN and the one provided by the UE (if any);
- [S13] EPS Bearer ID handling: creation and assignment of the identifier of the whole tunnel (from UE to P-GW);
- [S14] DRB ID UL handling: creation and assignment of the identifier related to the (control plane and data plane) tunnel between UE and eNB (upload);
- [S15] DRB ID DL handling: creation and assignment of the identifier related to the (control plane and data plane) tunnel between UE and eNB (download);
- [S16] S1 TEID UL handling: creation and assignment of the identifier related to the (data plane) tunnel between eNB and S-GW (upload);
- [S17] S1 TEID DL handling: creation and assignment of the identifier related to the (data plane) tunnel between eNB and S-GW (download);
- [S18] S5 TEID UL handling: creation and assignment of the identifier related to the (control plane and data plane) tunnel between S-GW and P-GW (upload);
- [S19] S5 TEID DL handling: creation and assignment of the identifier related to the (control plane and data plane) tunnel between S-GW and P-GW (download);
- [S20] HSS QoS profile request: retrieval of the QoS profile stored in the HSS;
- [S21] SPR QoS profile request: retrieval of the QoS profile stored in the SPR;
- [S22] Final QoS profile generation: production of the QoS values actually used in all the forwarding devices;
- [S23] TFT handling: creation and distribution of the TFT UL/DL;
- [S24] X2 TEID DL handling: creation and assignment of the identifier related to the (data plane) tunnel between SeNB and TeNB;
- [S25] Status snapshot: gathering of information related to the current situation about uplink and downlink packets;
- [S26] S1 eNB TEID DL: creation and assignment of the identifier related to the (data plane) tunnel between S-GW and TeNB;
- [S27] S1 S-GW TEID UL: creation and assignment of the identifier related to the (data plane) tunnel between SeNB and S-GW;
- [S28] Buffering of downlink packets - Idle state: temporary buffering of packets for the UE during Idle state (it is also included the notification service to inform the network about the presence of incoming packets);
- [S29] Mobility anchor: the entity which acts as the junction point between the different eNBs;
- [M30] Enforcement of QoS rules: application of the received QoS rules on the packet flows;
- [S31] Buffering of downlink packets - Handover: temporary buffering of downlink packets for the UE during handover procedure.

Network Element	**Tasks**
eNB	[S03], [S05], [S11], [S14], [S15], [S17], [S24], [S25], [S26], [M30], [S31]
MME	[S01], [S04], [S06], [S07], [S08], [S10], [S12], [S13], [S20]
S-GW	[S16], [S19], [S27], [S28], [S29], [M30]
PCRF	[S21], [S22]
P-GW	[S02], [S18], [S23], [M30]
HSS	[S09]
SPR	No tasks

Table 3.2: **Tasks assignment in the current LTE architecture.** *eNB, S-GW, and P-GW are in charge both of control plane and data plane tasks.*

Please notice that there is almost a perfect matching between the listed tasks and the whole set of information considered. In particular, there is a strong relationship between the creation of an information and the corresponding handling task. Therefore, we can identify the network element providing the handling functionality simply finding the network element that creates the corresponding state variable. Moreover, some information have not a corresponding handling task, but rather a distribution task. This happen especially for information that are already in the entities before the execution of any procedure (*e.g.* MME ID, IMSI, LTE K, etc.). Please notice that the UE is excluded from this analysis, since the subsequent refactoring process should not deal with the UE. Doing so, no changes are needed from the user-side in order to benefit of the proposed model.

Table 3.2 shows which elements of the current LTE architecture are in charge of which identified tasks. Please notice that some elements of the architecture carry out both control plane and data plane tasks. For example, the eNB takes care about low-level protocols variables (task [S03]) and enforces the QoS rules received (task [M30]), which are data plane tasks. At the same time, it manages its own tunnels' identifiers (tasks [S05], [S14], and many others) and chooses the cryptographic algorithm to adopt during the radio-level security procedure (task [S11]): all of them correspond to control plane tasks. Similarly, S-GW and P-GW carry out tasks belonging to different categories as well.

We argue that mixtures of control plane and data plane tasks are the first cause of the inflexibility affecting the current LTE architecture. Indeed, as long as the tasks of different categories are assigned to the same entity, the precise definition of the role of that entity becomes cumbersome. In this context, the application of the SDN paradigm consists in the first step of the following refactoring process.

Chapter 4

Refactoring Step 1: Separating Control and Data Planes

From the previous chapter, we know that the main reason for the inflexibility affecting the current LTE architecture is the mixture of control and data plane tasks carried out by every network element. In this context, every device results as a complex black-box because of its vaguely defined role. This particular condition leads to all the disadvantages we have already listed: each device is irreplaceable, non upgradable, difficult to maintain, and expensive.

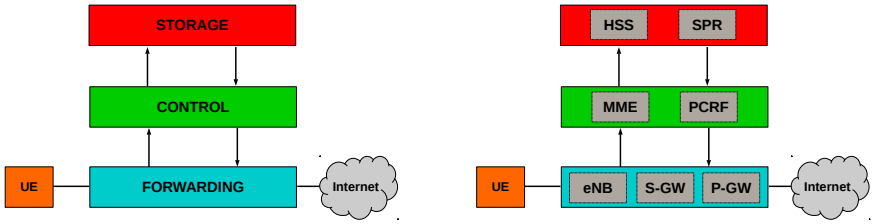
Therefore, the first step in the refactoring process is the recognition of categories of duties. Each category is a set of homogeneous tasks that identifies a precise role in the network. A network element that is mapped to a category executes only tasks that belong to the assigned category. In this scenario, a mapping between network elements and categories allows to gain automatically network elements whose roles are well-defined.

4.1 A Three-Levels Abstraction

The identification of the categories of duties is straightforward. The Software-Defined Networking paradigm already provides us two precise categories, which are control plane tasks and data plane tasks. In the context of LTE networks, examples of control plane tasks are the handling of the authentication and security procedures, or the tuning of the QoS parameters. Data plane tasks are instead the setting up of bearers for data transmissions and the enforcement of the QoS rules received.

Nevertheless, these two categories are not exhaustive. In fact some network elements provide customer-specific values in response to requests. HSS and SPR have this behaviour when MME and PCRF ask for authentication parameters and QoS profile values. We argue that these tasks do not belong neither to the control plane nor to the data plane. Instead, they consist in a separated group of tasks, which corresponds to database-like services.

Therefore, the categories of tasks allow us to abstract the LTE architecture in three layers:



(a) Three-layers abstraction. (b) Mapping with the LTE architecture.

Figure 4.1: **The three-layers abstraction and the correspondence with the elements of the current LTE architecture.** *The entities belonging to a level are supposed to execute tasks of that level only, and therefore the extra tasks have to be shifted to other entities, or to be executed by additional ones.*

- the forwarding layer, which consists in devices that execute data plane tasks;
- the control layer, which consists in devices that execute control plane tasks;
- the storage layer, which consists in devices that provide database-like services.

Figure 4.1a shows how each layer relate with the other layers, with the UE, and with the rest of the network. The forwarding layer acts as the data bridge between the UE and the Internet. It informs the overlying control layer about all the events that occur in the network (*e.g.* UE exiting from the coverage area of the antenna, attach request by the UE). The control layer takes decisions according to the notifications received from the underlying forwarding layer, and consequently instructs the forwarding devices on how to behave. Moreover, in order to begin some procedures, the control layer needs customer-specific values, which are stored in a database-like structure. Therefore, the control layer contacts the overlying storage layer in order to retrieve the desired information.

The next step is the mapping between the elements of the current LTE architecture and the layers that we have identified. We already know that HSS and SPR are mapped to the storage layer. The MME is the real brain of the network, since it is in charge of taking decisions using the information provided by eNB and S-GW. Therefore, the MME is mapped to the control layer. Similarly, the PCRF decides the QoS policy and it instructs the P-GW accordingly, and therefore it is mapped to the control layer as well. The aim of the remaining devices (eNB, S-GW and P-GW) is to carry data from the UE to the Internet and vice-versa. Since this is a data plane function, the devices are mapped to the forwarding layer. The complete mapping is shown in Figure 4.1b.

Every device is supposed to carry out only tasks that belong to the assigned layer, but we know that the network elements of the LTE architecture execute tasks belonging to different layers. In order to solve this discrepancy, the tasks that do not fall in the assigned layer have to be executed by devices belonging to their own layer. In this refactoring step, every time a device has some of these extra tasks, we create an additional ad-hoc entity in the layer to which the extra tasks belong, and the extra tasks are assigned to it.

4.2 The Refactoring Principles

A clear aim and a set of rules are the ingredients for a beneficial refactoring process. The objective of the refactoring step is the clear separation of the duties in every architectural element, but we still lack the set of principles that drives the process. We have carried out the refactoring step following some guidelines, which allowed us to motivate the choice performed at every decision point. The list of the guidelines follows.

- *Never touch the interface with the UE.* By definition, refactoring means changing the internal keeping untouched the external. Therefore, the way in which the UE and the cellular network communicate has to be kept as it is. There is no point in proposing an architecture that requires the substitution or the upgrade of all 4G-enabled mobile phones, since it terribly lacks of feasibility.

- *Never touch the security procedures.* Since the interface with the UE has not to be touched, and since the UE uses some standard algorithms during authentication and security parameters exchange, the algorithms have not to be changed on the network side as well.

- *For every device belonging to the forwarding layer, create its personal control entity and shift all the control tasks to it.* The aim of the rule is to make the forwarding layer as dumb as possible, limiting its functions to the enforcement of the rules received and to the notification of network events.

The rules express exactly the final aim of the refactoring step: keep the services provided to the UE as they are, and stratify the internal structure in the three proposed layers. Section 4.3 shows the outcome of the process.

4.3 The Proposed Model

Figure 4.2 depicts the outcome of the first refactoring step. We have split each one of the forwarding entities in two sub-entities: the data plane part, in the forwarding layer, and the control plane part, in the control layer. A communication channel links the two parts, such that notifications and instructions can reach their target directly. The interface of this communication channel starts with the initial letter of the split entity, and it ends with "CD", in order to highlight that it links control and data planes. Please notice that we had modified some of the pre-existent links according to the introduced separations. For example, in the current LTE architecture, a control plane communication channel links eNB and MME, and a data plane communication channel links the eNB with the S-GW. Therefore, we put a data plane channel linking the data plane parts of the eNB and of the S-GW, and we put a the control plane channel between the control plane part of the eNB and the MME. Essentially, the rule we adopted is to transfer an original control plane channel to the control layer, and to transfer an original data plane channel to the forwarding layer. If two entities use a same channel both for control and data planes, then two channels are created, one in the control layer linking the control plane parts, and one in the forwarding layer linking the data plane parts (*e.g.* S-GW and P-GW).

We have also put HSS and SPR in a single entity, which represents the storage layer. MME and PCRF still access to the storage layer using their original interfaces (*i.e.* S6a and Sp), even if the device is now the same. Please notice that HSS and

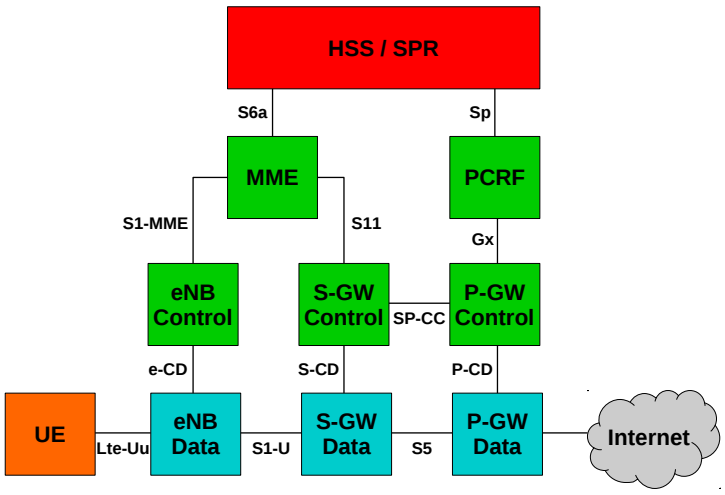


Figure 4.2: **The model after the first refactoring step.** *eNB, S-GW and P-GW are split in control plane and data plane parts, while the storage layer consists in a single database which comprises both HSS and SPR.*

SPR still execute their control plane tasks as the refactoring step is focused on the separation between control and data planes.

4.4 The Network Events

We have modified the procedures triggered by the considered network events according to the proposed model. In the following, we present the most significant changes for each one of them. The reader can find the details of all the modified procedures in [101].

4.4.1 Initial Attach

The new procedure requires a number of signals which is much greater than the number requested by the original procedure. UE and eNB exchange a significant amount of signals, especially in the first steps of the procedure, and since now the eNB consists in two parts, a message arriving to or departing from the eNB automatically implies two messages in the proposed model. Moreover, the NAS security establishment requires an exchange of information between UE and MME in the original procedure, and therefore eNB acts as a simple bypass for the signals; in the modified procedure, the bypass is through two entities as the eNB control and data parts substitute the original eNB.

The impact on the signaling load coming from the splitting of S-GW and P-GW is small, as the number of signals related with these two entities is narrow also in the original procedure. Even if it seems that the refactoring has brought just disadvantages, if we look the whole procedure, we can notice that the separation between control and data plane is clear. The control plane part of each entity communicates

with the data plane part providing instructions, while the data plane part's signals directed to the corresponding control part are only notifications or acknowledgements.

The whole procedure is extensively described in [99].

4.4.2 Active_to_Idle Transition

Similarly to the previous situation, the procedure for the new model is heavier than the original procedure, as it requires almost the double of the signals. Since eNB and S-GW are the main players in the procedure, this effect is obvious.

Nevertheless, the new procedure shows the relationship between MME and the control entities of eNB and S-GW: the MME acts as the master, and the specific control entities are the slaves, in the sense that everything revolves around the MME. After the notification for the inactivity timeout, the eNB control notifies in turn the MME, which starts to handle concretely the event. Similarly, the MME instructs the S-GW control to close the downlink endpoint of the data bearer, which instructs in turn the S-GW data. This hierarchical relationship comes naturally as MME controls partially eNB and S-GW in the original LTE architecture, and therefore MME controls partially the control parts of eNB and S-GW in the new architecture. We argue that this hierarchical relationship is useless, as the control layer should be flat, in order to save signals that are internal to the same layer. This consideration drove the second refactoring step.

The whole procedure is extensively described in [96].

4.4.3 Idle_to_Active Transition

The following sections describe the differences between the procedure in the new model and the original procedure in case of UE transiting from idle to active state. As usual, the procedures are different depending on the cause that triggered the procedure (*i.e.* the UE needs to upload packets, or the network has some downlink packets for the UE).

4.4.3.1 UE Triggered

The procedure is similar to the initial attach since the main aim is the same, which is establishing an active session. Therefore, we have the same pro and cons as outcome of the comparison. The first steps of the original procedure consist in an intense signals exchange between UE and eNB, which automatically implies a high increase in the signaling load of the new procedure. Similarly, when the MME instructs the S-GW to open the downlink endpoint of the data bearer, the signaling load of the new procedure is heavier because of the indirection between S-GW control and data parts. The hierarchical relationship between the control entities is less evident in this comparison, since the eNB control is able to handle the greatest part of the session re-establishment without interacting with the MME.

The whole procedure is extensively described in [98].

4.4.3.2 Network Triggered

The procedure triggered by the network equals to the procedure triggered by the UE, except for few additional steps at the beginning. In the procedure of the original LTE architecture, the notification of incoming downlink packets flows from the S-GW to the UE, requiring four additional signals. The procedure of the new model requires

six additional signals, because of the division of the S-GW (one signal more) and the division of the eNB (one signal more).

The whole procedure is extensively described in [97].

4.4.4 Handover

The following sections describe the comparison between the procedures triggered by an handover event in the proposed model and in the original LTE architecture. The way in which the handover is performed (*i.e.* through the X2 interface or not) determines the procedure triggered, therefore we report here a comparisons for each one of the possible procedures.

4.4.4.1 X2 Handover

The procedure of the original LTE architecture comprises a lot of messages exchanged on the X2 interface between the source eNB (SeNB) and the target eNB (TeNB), especially in the first steps because of the UE's context transfer. Similarly to the S5 interface between S-GW and P-GW, the two eNBs use the X2 interface both for control plane and data plane messages, which is not a proper usage. In the proposed model, the X2 interface links only the data parts of the two eNBs, while the control parts use another separated communication channel. On one hand, this brings architectural advantages since it consists in a clear separation between control and data planes; on the other hand, the addition of entities to the architecture implies higher signaling overhead. Looking at the whole procedure, it requires 16 signals more than the original version, which is a significant drawback. The tight interaction between the two eNBs consists in the main source of overhead, as the handling of the event in the rest of the network is simple in the procedure of the new model as well.

The whole procedure is extensively described in [102].

4.4.4.2 S1 Handover

Since the lack of a linking interface between the two eNBs triggers the S1 handover procedure, this automatically implies that all the messages between the two eNBs have to flow through a common junction point. In particular, the MME is the junction point for the control messages, and the S-GW is the junction point for the data messages. If the S1 handover triggers a signaling-demanding procedure already in the original LTE architecture, the procedure in the new model is even worse, mostly because of all the separations that play an important role in the signaling increase (source eNB, target eNB, and S-GW separations). This is evident in the very first steps of the procedure, when the control parts of the two eNBs communicate using the MME as intermediary and instruct the data part using the data received from the other eNB. The scenario is exactly the same of all the previous procedures: the drawback of the architectural improvements brought by SDN is the additional overhead in terms of signaling load.

Moreover, the S1 handover procedure of the proposed model is a glaring example of the hierarchical relationship between the control entities, as the MME coordinates every step of the procedure. The control entities always send an acknowledgement to the MME (possibly in piggyback) after the execution of an action: we argue that these acknowledgement are useless because they are internal to the same layer, and therefore a compression of the control entities would be beneficial. That is exactly the direction taken during the second refactoring step.

The whole procedure is extensively described in [100].

4.5 Signaling Load Analysis

In this section, we provide some data that allow to evaluate the proposed model. First, we give the equations that compute the total time requested by each procedure. Second, we summarizes the results of the comparison with the current LTE architecture on the signals required in handling the network events.

4.5.1 Interfaces

We assigned a variable to each interface used in the proposed model. Each variable represents the time requested by a signal exchange on that interface. The union between these variables and the frequency with which each interface has been used allows us to define the total time requested by the procedure of each event through a simple equation. The list of variables follows:

- $c1$ corresponds to the cost associated to interface LTE-Uu, between UE and eNB Data;
- $c2$ corresponds to the cost associated to interface S1-U, between eNB Data and S-GW Data;
- $c3$ corresponds to the cost associated to interface S5, between S-GW Data and P-GW Data;
- $c4$ corresponds to the cost associated to interface S1-MME, between eNB Control and MME;
- $c5$ corresponds to the cost associated to interface S11, between S-GW Control and MME;
- $c6$ corresponds to the cost associated to interface Gx, between P-GW Control and PCRF;
- $c7$ corresponds to the cost associated to interface S6a, between MME and HSS;
- $c8$ corresponds to the cost associated to interface Sp, between PCRF and SPR;
- $c9$ corresponds to the cost associated to interface X2, between two eNB Data parts (if the eNBs are linked);
- $c10$ corresponds to the cost associated to interface e-CD, between eNB Control and eNB Data;
- $c11$ corresponds to the cost associated to interface S-CD, between S-GW Control and S-GW Data;
- $c12$ corresponds to the cost associated to interface P-CD, between P-GW Control and P-GW Data;
- $c13$ corresponds to the cost associated to interface SP-CC, between S-GW Control and P-GW Control;
- $c14$ corresponds to the cost associated to interface ee-CC, between two eNB Control parts (if the eNBs are linked).

4.5.2 Initial Attach

The list of messages exchanged during Initial Attach follows:

- 13 messages exchanged over LTE-Uu interface (Steps 1-4-5-8-9-16-17-22-23-44-45-48-51);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 8 messages exchanged over S1-MME interface (Steps 11-14-19-20-25-42-50-53);
- 4 messages exchanged over S11 interface (Steps 28-41-54-57);
- 2 messages exchanged over Gx interface (Steps 32-35);
- 4 messages exchanged over S6a interface (Steps 12-13-26-27);
- 2 messages exchanged over Sp interface (Steps 33-34);
- 0 messages exchanged over X2 interface;
- 14 messages exchanged over e-CD interface (Steps 2-3-6-7-10-15-18-21-24-43-46-47-49-52);
- 6 messages exchanged over S-CD interface (Steps 29-30-39-40-55-56);
- 2 messages exchanged over P-CD interface (Steps 36-37);
- 2 messages exchanged over SP-CC interface (Steps 31-38);
- 0 messages exchanged over ee-CC interface.

The total amount of messages required by the procedure is 57. The total amount of time requested by the message exchanging in the procedure may be expressed as:

$tot\_time = 13*c1+8*c4+4*c5+2*c6+4*c7+2*c8+14*c10+6*c11+2*c12+2*c13$

4.5.3 Active_to_Idle Transition

The list of messages exchanged during Active_to_Idle transition follows:

- 1 message exchanged over LTE-Uu interface (Step 9);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 3 messages exchanged over S1-MME interface (Steps 2-7-11);
- 2 messages exchanged over S11 interface (Steps 3-6);
- 0 messages exchanged over Gx interface;
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over Sp interface;
- 0 messages exchanged over X2 interface;

- 3 messages exchanged over e-CD interface (Steps 1-8-10);
- 2 messages exchanged over S-CD interface (Steps 4-5);
- 0 messages exchanged over P-CD interface;
- 0 messages exchanged over SP-CC interface;
- 0 messages exchanged over ee-CC interface.

The total amount of messages required by the procedure is 11. The total amount of time requested by the message exchanging in the procedure may be expressed as:

$$tot\_time = c1 + 3 * c4 + 2 * c5 + 3 * c10 + 2 * c11$$

4.5.4 Idle_to_Active Transition

4.5.4.1 UE Triggered

The list of messages exchanged during Idle_to_Active transition - UE triggered follows:

- 8 messages exchanged over LTE-Uu interface (Steps 1-4-5-8-9-14-15-18);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 3 messages exchanged over S1-MME interface (Steps 11-12-20);
- 2 messages exchanged over S11 interface (Steps 21-24);
- 0 messages exchanged over Gx interface;
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over Sp interface;
- 0 messages exchanged over X2 interface;
- 9 messages exchanged over e-CD interface (Steps 2-3-6-7-10-13-16-17-19);
- 2 messages exchanged over S-CD interface (Steps 22-23);
- 0 messages exchanged over P-CD interface;
- 0 messages exchanged over SP-CC interface;
- 0 messages exchanged over ee-CC interface.

The total amount of messages required by the procedure is 24. The total amount of time requested by the message exchanging in the procedure may be expressed as:

$$tot\_time = 8 * c1 + 3 * c4 + 2 * c5 + 9 * c10 + 2 * c11$$

4.5.4.2 Network Triggered

The list of messages exchanged during Idle_to_Active transition - Network triggered follows:

- 9 messages exchanged over LTE-Uu interface (Steps 6-7-10-11-14-15-20-21-24);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 4 messages exchanged over S1-MME interface (Steps 4-17-18-26);
- 4 messages exchanged over S11 interface (Steps 2-3-27-30);
- 0 messages exchanged over Gx interface;
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over Sp interface;
- 0 messages exchanged over X2 interface;
- 10 messages exchanged over e-CD interface (Steps 5-8-9-12-13-16-19-22-23-25);
- 3 messages exchanged over S-CD interface (Steps 1-28-29);
- 0 messages exchanged over P-CD interface;
- 0 messages exchanged over SP-CC interface;
- 0 messages exchanged over ee-CC interface.

The total amount of messages required by the procedure is 30. The total amount of time requested by the message exchanging in the procedure may be expressed as:

$$tot\_time = 9 * c1 + 4 * c4 + 4 * c5 + 10 * c10 + 3 * c11$$

4.5.5 Handover

4.5.5.1 X2 Handover

The list of messages exchanged during X2 handover follows:

- 3 messages exchanged over LTE-Uu interface (Steps 1-8-13);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 2 messages exchanged over S1-MME interface (Steps 17-26);
- 2 messages exchanged over S11 interface (Steps 18-25);
- 2 messages exchanged over Gx interface (Steps 22-23);
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over Sp interface;

- 0 messages exchanged over X2 interface;
- 14 messages exchanged over e-CD interface (Steps 2-4-5-7-9-11-12-14-15-16-28-29-30-31);
- 2 messages exchanged over S-CD interface (Steps 19-20);
- 0 messages exchanged over P-CD interface;
- 2 messages exchanged over SP-CC interface (Steps 21-24);
- 4 messages exchanged over ee-CC interface (Steps 3-6-10-27).

The total amount of messages required by the procedure is 31. The total amount of time requested by the message exchanging in the procedure may be expressed as:

$$tot\_time = 3 * c1 + 2 * c4 + 2 * c5 + 2 * c6 + 14 * c10 + 2 * c11 + 2 * c13 + 4 * c14$$

4.5.5.2 S1 Handover

The list of messages exchanged during S1 handover follows:

- 3 messages exchanged over LTE-Uu interface (Steps 1-14-20);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 9 messages exchanged over S1-MME interface (Steps 3-4-7-12-16-17-22-31-34);
- 6 messages exchanged over S11 interface (Steps 8-11-23-30-35-38);
- 2 messages exchanged over Gx interface (Steps 27-28);
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over Sp interface;
- 0 messages exchanged over X2 interface;
- 13 messages exchanged over e-CD interface (Steps 2-5-6-13-15-18-19-21-32-33-39-40-41);
- 6 messages exchanged over S-CD interface (Steps 9-10-24-25-36-37);
- 0 messages exchanged over P-CD interface;
- 2 messages exchanged over SP-CC interface (Steps 26-29);
- 0 messages exchanged over ee-CC interface.

The total amount of messages required by the procedure is 41. The total amount of time requested by the message exchanging in the procedure may be expressed as:

$$tot\_time = 3 * c1 + 9 * c4 + 6 * c5 + 2 * c6 + 13 * c10 + 6 * c11 + 2 * c13$$

4.5.6 Summary

Table 4.1 summarizes the results from the previous sections.

	IA	AtI	ItA (UE)	ItA (Net)	X2H	S1H	Total
LTE-Uu	13	1	8	9	3	3	37
S1-U	0	0	0	0	0	0	0
S5	0	0	0	0	0	0	0
S1-MME	8	3	3	4	2	9	29
S11	4	2	2	4	2	6	20
Gx	2	0	0	0	2	2	6
S6a	4	0	0	0	0	0	4
Sp	2	0	0	0	0	0	2
X2	0	0	0	0	0	0	0
e-CD	14	3	9	10	14	13	63
S-CD	6	2	2	3	2	6	21
P-CD	2	0	0	0	0	0	2
SP-CC	2	0	0	0	2	2	6
ee-C	0	0	0	0	4	0	4
Total	57	11	24	30	31	41	

Table 4.1: **Signaling load analysis of the first proposed model.** *S5 and X2 interfaces have no messages as they are data plane interfaces, but their load is shifted on the interfaces linking the control plane parts of the elements.*

Table 4.2 shows the difference in the number of messages with the previous model instead.

	IA	AtI	ItA (UE)	ItA (Net)	X2H	S1H	Total
LTE-Uu	0	0	0	0	0	0	0
S1-U	0	0	0	0	0	0	0
S5	-2	0	0	0	-2	-2	-6
S1-MME	0	0	0	0	0	0	0
S11	0	0	0	0	0	0	0
Gx	0	0	0	0	0	0	0
S6a	0	0	0	0	0	0	0
Sp	0	0	0	0	0	0	0
X2	0	0	0	0	-4	0	-4
e-CD	+14	+3	+9	+10	+14	+13	+63
S-CD	+6	+2	+2	+3	+2	+6	+21
P-CD	+2	0	0	0	0	0	+2
SP-CC	+2	0	0	0	+2	+2	+6
ee-C	0	0	0	0	+4	0	+4
Total	+22	+5	+11	+13	+16	+19	

Table 4.2: **Comparison between the current LTE architecture and the first proposed model.** *The small reductions on S1 and X2 interfaces cannot compensate the huge amount of signals required on the newly introduced interfaces.*

The scenario depicted by the previous sections describe a deplorable situation, as

Network Element	Tasks
eNB Control	[S03], [S05], [S11], [S14], [S15], [S17], [S24], [S26]
eNB Data	[S25], [M30], [S31]
MME	[S01], [S04], [S06], [S07], [S08], [S10], [S12], [S13], [S20]
S-GW Control	[S16], [S19], [S27]
S-GW Data	[S28], [S29], [M30]
PCRF	[S21], [S22]
P-GW Control	[S02], [S18], [S23]
P-GW Data	[M30]
HSS	[S09]
SPR	No tasks

Table 4.3: **Tasks assignment after the first refactoring step.** *The proposed model splits the tasks of eNB, S-GW and P-GW between their control and data components.*

the signaling load of the new model is much worse than the one of the original LTE architecture. The price of the separation between control and data plane is therefore very high. Nevertheless, in this refactoring step we have performed a dumb separation, simply focusing on the application of the SDN principles and ignoring every efficiency consideration. Therefore, the actual reduction of the signal load consists in one of the main objective of the next refactoring step.

4.6 The Network Functions

Table 4.3 shows the allocation of the network functions after the first refactoring step. The assignment is similar to the original one, as most of the entities still handle the same set of task they execute in the current LTE architecture. In fact, the separation between control and data planes touches eNB, S-GW, and P-GW, and they are the only entities whose tasks list changes.

The procedure for gaining the new assignment is straightforward. Essentially, for each one of the touched entities, the new model cuts the original task list in two lists: one of the control plane tasks, and the second of the data plane tasks. Then, the control plane part of the entity executes the list of control plane tasks, and the data plane part executes the list of data plane tasks.

We can see that the number of tasks of the data plane parts is generally very small, and the common task among all the data plane parts is the enforcement of the QoS rules received. This is a beneficial outcome: the less tasks the data plane parts have to execute, the more dumb they are. In the context of SDN, being dumb implies uniformity, interchangeability, affordability. The first refactoring step provides an architecture in which the forwarding elements share a significant amount of common characteristics: they enforce QoS rules received, they notifies the control layer of the network events, and they are instructed by it on how to behave. This consists in the first concrete step in the direction of flexibility.

Chapter 5

Refactoring Step 2: Reducing the Control Entities

The software-defined model brought by the first refactoring step, even if beneficial from the architectural perspective, comes with an excessive signaling overhead because of the splitting of eNB, S-GW, and P-GW in control and data parts. From the signaling storms' point of view, we have just worsen an already complicated situation. This chapter describes the second refactoring step, whose aim is the reduction of the signaling load introduced by the proposed model, but keeping the separation between control and data planes.

5.1 An Excessive Number of Entities

The signaling load analysis of the previous chapter has depicted a distressing scenario. In fact, the simple separation of control and data planes in eNB, S-GW, and P-GW has the side effect of adding a signal almost every time a communication includes one of them. Indeed, if a signal reaches the data plane part, a notification signal goes to the control plane part, and every time a signal reaches the control plane part, an instruction reaches the data plane part. Even if this seems the motivation of the issue, we actually want this kind of interface between the control plane and the data plane, and we have focused the previous refactoring step exactly on this. Therefore, we have to solve the issue from another perspective.

During the previous refactoring step, we have pointed out that the hierarchical relationship that holds between the control entities is actually useless, since it implies signals between entities belonging to the same layer. In fact, the direction of the refactoring process is to keep all the hardware requirements on the forwarding layer, and move all the software requirements on the control layer. When most of the software requirements are in the control layer, a single entity is potentially enough for executing all the tasks of the layer.

We make the things clear through an example. Consider the eNB of the original LTE architecture. The first refactoring step has separated its control plane form its data plane. This means that an external entity (*i.e.* the eNB control) executes the control tasks, while the physical eNB (*i.e.* the eNB data) takes care only about sending notifications to the control layer, enforcing rules received, and dealing with radio-level protocols. The outcome of the process is the following: the external entity has most

of the software requirements, since it takes care about the control tasks, but it has no hardware requirements, since a normal remote server can execute them. Instead, the eNB has very limited software requirements, since it has to offer a simple software-defined interface and to take care about the lower level protocols, but it has all the hardware requirements, since we actually need a physical eNB in order to communicate with the UE (*i.e.* it has the right radio interface).

Now consider the MME, which has a lot of software requirements, since it cares about most of the control plane tasks, and no hardware requirements, since there are no required physical properties in the specifications of the MME. In this context, there is no point in keeping the eNB control separated from the MME, as they are two merely-software entities, and therefore we can put them together in the same physical device (*i.e.* the remote server). We can make similar considerations for PCRF, S-GW Control and P-GW Control of course. The compression of the control entities has a twofold advantage: it allows to reduce significantly the amount of signals needed in the procedures, and it keeps the software-defined interface designed during the first refactoring step.

There are many other works that support the idea of coalescing the control entities in a single physical device. A centralized controller improves the management of the base stations, which can be configured to avoid interferences and optimize power consumptions [77, 65]. Since we have a single managing device, we can significantly lower the OPerating EXpenditure (OPEX) of the cellular network as well. The work in [60] promotes the shifting of the RAN control entities to the cloud, which implies a logically centralized controller on remote servers. The cloud RAN allows resource pooling and scalability, as the resources (*i.e.* base stations) can serve multiple tenants through dynamic configurations. In fact, a single view of the whole network allows to have the same base stations used by several mobile operators contemporarily: in this context, the controller decides the resource assignment policy taking into account operators' needs and network status (*e.g.* transient loads, flash crowds, and so on). These studies show that the advantages brought by the centralization of the controller are not limited to the reduction of the signaling load, but we gain some important improvements from the network management perspective as well.

5.2 The Refactoring Principles

During the second refactoring step, we have kept the same refactoring principles adopted during the first step. In fact, we actually performed the second step starting from the output of the first one, and therefore the principles adopted in the first step are still valid. Nevertheless, we have integrated the original list with some additional rules:

- *Represent eNB Control, MME, and PCRF as a single entity.* The aim of the rule is to ensure efficiency, since the effect is a significant reduction in the number of signals requested during the procedures considered.
- *Represent S-GW Control and P-GW Control as a single entity.* The objective is the same of the previous guideline. No messages are needed between them if they are compressed in the same physical device.
- *Shift any control task executed by the storage layer to the control layer.* While the first refactoring step was focused just on the separation between control and data planes, now we want to complete the stratification of the architecture. The

rule ensures that the devices belonging to the storage level take care to provide database-like services only.

It could seem strange that we have coalesced the control layer in two entities instead of a single one. The reason behind is to keep a separation between the Radio Access Network (RAN) and the Core network, as they are two modular, independent parts. If the Core control is separated from the RAN control, it means that in the future we could link the same core network to another RAN, simply configuring the link between the Core Control and the new RAN Control. Moreover, having a single control entity exposes to the risk of single point of failure: in the global controller goes down, the whole network will not be working.

A secondary rule we have adopted in this step is to re-shift some minor control tasks from the eNB Control back to the eNB Data. This tasks consist in the handling of radio-level procedures (*e.g.* RRC connections) and radio-level identifiers (*e.g.* GUTI). Even if this could seem illogical, this is the best compromise between architectural benefits and efficiency. In fact, we have noticed that in the previous model the eNB Data acts as a by-pass between the UE and the eNB Control too many times, impacting significantly the amount of signals requested. We argue that it is possible to keep the radio-level minor tasks inside the eNB Data itself without damaging the software-defined interface designed during the first refactoring step. This choice allows also to avoid the excessive latencies that would come from carrying out these protocols in the control layer [60, 65]. In fact, as the times required to execute locally a procedure and to send a signal on a communication channel are not comparable, from the performance perspective it is better to avoid a tight interaction between control and forwarding layers every time a radio-level protocol needs to be executed.

5.3 The Proposed Model

Figure 5.1 shows the output of the second refactoring step. Following the refactoring principles, we have created two main control entities: the RAN Control and the Core Control. The RAN Control includes the handling of the eNB Data, the MME, and the PCRF, and therefore it is in charge of almost all the decision tasks of the network. In fact, the Core Control has simply to apply the QoS rules received by the RAN Control, or to instruct S-GW and P-GW according to the commands received by the RAN Control. In other words, the only decision tasks that are in charge of the Core Control are the handling of the tunnel identifiers in S-GW and P-GW.

It could seem that a hierarchical relationship still holds among the control plane entities, as the real brain is the RAN Control, and the Core Control mostly takes orders from it. That is true, but, as we have already explained, this is necessary to gain a modular architecture, that separates the handling on the RAN from the handling of the Core network. Please notice that we have removed all the previous newly introduced interfaces between the control entities, as they are no more requested. We have only added a single interface linking the RAN Control to the Core Control (RC-CC).

The schema shows the interface linking the S-GW to the P-GW as a dashed line. The idea behind is that the path from the S-GW and the P-GW could be changeable, constituting of a direct link between them, or of a chain of intermediary gateways of arbitrary length. In fact, we have noticed that S-GW and P-GW actually share a lot of properties, as they are both gateways: they mostly have to take care of data bearers and to apply QoS rules. The S-GW has the only particularity of being the

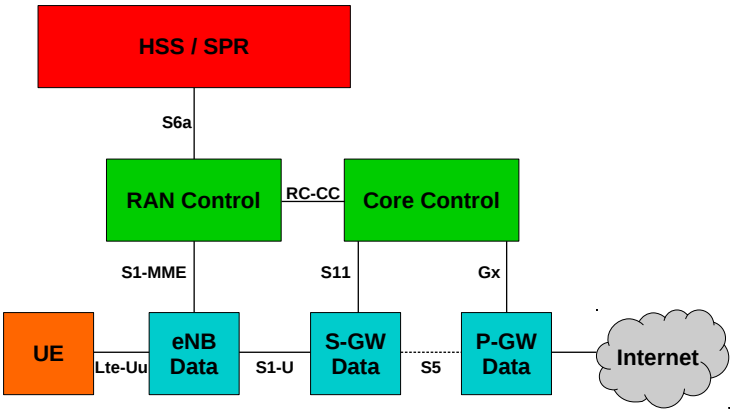


Figure 5.1: **The model after the second refactoring step.** *We have coalesced eNB Control, MME and PCRF in RAN Control, and S-GW Control and P-GW Control in Core Control. S-GW Control and P-GW Control do not need directly any customer-specific information, and therefore a link between Core Control and storage layer is unnecessary.*

first junction point between eNBs, and the P-GW has the only particularity of being the last gateway before exiting the cellular network domain. Once we have these two characteristics in two devices, then we can act between them as we prefer, since every device in the middle has essentially to take care of the enforcement of the QoS rules received only.

Figure 5.2 depicts an example of dynamic configuration of the data path in the core network. The initial scenario assumes that the links between gateway G1 and the base stations (S1 and S2) are not available. In this context, gateway G2 acts as the first anchor point between the base stations, and therefore it consists in the S-GW of the core network. Once the links between G1 and the base stations become available, the first junction point becomes gateway G1. Therefore, the Core Control shifts the S-GW role to gateway G1, and gateway G2 becomes a simple intermediary gateway which will have only to enforce the QoS rules received by the Core Control on the data flows.

Finally, we have put a single interface linking the storage layer with the control layer, instead of the previous two. This is a natural simplification as the only entities that require to be connected to the storage layer are MME and PCRF, and they have been coalesced in the same entity, the RAN Control. Therefore, a single link between RAN Control and the storage layer is definitively enough.

5.4 The Network Events

As in the previous refactoring step, we report here the main differences between the procedures adopted in the new proposed model and the original LTE procedures. The

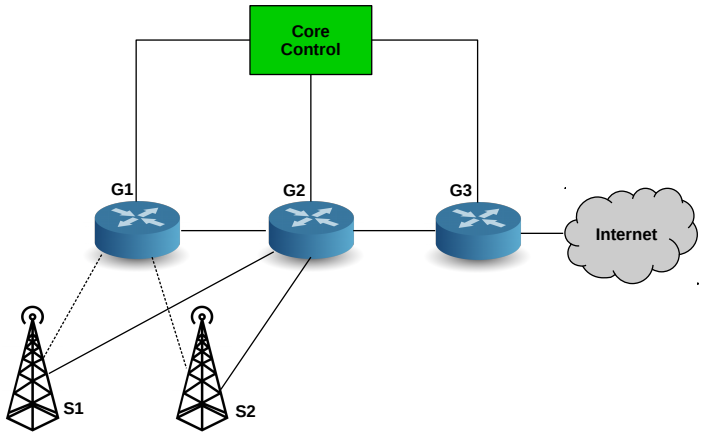


Figure 5.2: **Elasticity example in the core network.** *As the links between the base stations and the gateway G1 become available, the Core Control shifts the S-GW role from G2 to G1.*

reader can find the details of all the modified procedures in [108].

5.4.1 Initial Attach

We have performed a lot of improvements in the initial attach procedure, which lead the number of signals required to be significantly reduced. The most important advancement consists in the reduction of the accesses to the storage layer. In fact, the HSS is among the components that suffer most during signaling storms [121], and therefore a reduction in the accesses to the storage layer is beneficial. The storage layer is normally accessed three times during the initial attach procedure: the first time for retrieving customer-specific authentication values, and the other two times to retrieve the QoS profile values from HSS and SPR. Since we have coalesced MME and PCRF in the same entity, there is no point in accessing the storage layer in different moments. Therefore, when the RAN Control contacts the storage layer, the answer does not contain the authentication values only, but also the QoS profile of the considered customer.

The other considerable improvement is given in the simplification of the whole process. In fact, the original procedure has a convoluted sequence of steps: once the NAS security operations are concluded, the MME retrieves the QoS profile from the HSS and sends it to the PCRF (through S-GW and P-GW in sequence), which then retrieves another set of values from the storage layer and creates the final QoS rules, sending them back (from P-GW, to S-GW, and to eNB) in order to set up the data path. This is surely a contorted and illogical behaviour. In the procedure of the proposed model instead, the storage level is accessed one time for retrieving all the needed information, the QoS rules are generated directly in the RAN Control, which keeps them for instructing the eNB and sends them to the Core Control as well. The Core Control then instructs the gateways with the rules received.

The whole procedure is extensively described in [106].

5.4.2 Active_to_Idle Transition

Comparing the procedures, we can notice that the original LTE procedure is still more efficient than the one designed for the proposed model, even if the difference in performance is really small. The separation between the RAN Control and the Core Control is the source of the overhead, as the Core Control acts as intermediary between the RAN Control and the S-GW. In fact, in the current LTE architecture, the MME notifies directly the S-GW of the event, which closes the data bearer and sends back the acknowledgement to the MME. In the procedure of the new model, the RAN Control instructs the Core Control, which instructs in turn the S-GW, and then the acknowledgement follows the two-hops path back. Therefore the overhead is of only two signals more.

The whole procedure is extensively described in [103].

5.4.3 Idle_to_Active Transition

Similarly to the previous chapter, we report here the differences in the procedure adopted to make the UE transiting from Idle to Active state. Depending on the triggering source, the procedure adopted is different. In the following, we describe the differences for each one of the procedures.

5.4.3.1 UE Triggered

The situation is exactly the same of the Active_to_Idle transition. The procedure of the new model requires a small amount of additional signals because of the separation between RAN Control and Core Control. In fact, when the data bearer in the S-GW has to be reactivated, in the procedure of the new model the command goes from the RAN Control to the Core Control, and from the Core Control to the S-GW. Then, the acknowledgement flows back using the same reverse path. In the original LTE architecture, the MME talks directly with the S-GW, and therefore there is no indirection and no additional signals are requested. Please notice that shifting back the radio-level tasks to the eNB Data allows to limit the overhead of the new procedure.

The whole procedure is extensively described in [105].

5.4.3.2 Network Triggered

There are no significant differences with the comparison on the UE-triggered procedure, since the network-triggered one requires four signals more in order to let the S-GW notification of incoming traffic reaching the UE. The original LTE procedure requires four signals more as well, and therefore the overhead of the proposed procedure is the same as before.

It could seem strange that the amount of signals requested by the two compared procedures for the notification of the UE is exactly the same. In fact, we know that the Core Control in the proposed model consists in an additional indirection, and therefore we expect a message flowing from the S-GW to the UE to require a signal more. Nevertheless, the procedure of the original LTE architecture includes an additional message from the MME to the S-GW, whose aim is to acknowledge the reception of the notification. We argue that this message is actually useless, since the acknowledgement can be delivered in piggyback when the MME (*i.e.* the RAN

Control) gives the first instruction to the Core Control, which will do the same with the S-GW. Therefore, the number of additional signals requested by the two procedures is the same.

The whole procedure is extensively described in [104].

5.4.4 Handover

In the following, we report the differences in the two procedures used to handle handovers.

5.4.4.1 X2 Handover

Two conflicting trends influence the comparison between the two procedures. On one hand, the procedure of the new model does not allow control plane communications over the X2 interface, and therefore the RAN Control has to act as intermediary between the two eNBs, increasing the signaling demand. On the other hand, the new model manages the core network in a simpler way, and therefore the signals needed to notify the core about the event are less than before. Nevertheless, the comparison shows that there is almost equality in the number of signals requested, since the procedure of the new model asks for a single signal more.

The whole procedure is extensively described in [109].

5.4.4.2 S1 Handover

The original S1 handover procedure includes the creation of an indirect data tunnel, in order to bring the data reaching the source eNB (SeNB) to the target eNB (TeNB). Essentially, the data tunnel consists of three parts: the original bearer, which brings the data from the S-GW to the SeNB; a data tunnel that brings back the data from the SeNB to the S-GW; a data tunnel that brings the data from the S-GW to the TeNB. In this configuration, the data is sent back and forth over the same network link (*i.e.* the S1-U interface between SeNB and S-GW), an inefficiency which is known in the world of telecommunications with the name of 'trombone effect' [60]. We argue that this indirection is useless and expensive, since it would be smarter to avoid the S-GW - SeNB loop. Therefore, we designed a procedure which stops the forwarding of the packets to the S-GW, and makes the packets flowing again only when the tunnel with the TeNB is ready-to-use. Our design requires buffering capabilities on the gateway acting as S-GW; nevertheless, we argue that this requirement is close to what gateways are already able to offer, and therefore it does not consist in an obstacle during implementation.

The idea in the procedure of the proposed model follows. When the SeNB notifies the RAN Control about the imminent change of base station of the UE, the RAN control instructs the Core Control to start the buffering of the downlink packets, which instructs in turn the S-GW. Then, when the TeNB finishes the setup, it notifies the RAN Control about its availability to accept downlink packets. Therefore, the RAN Control instructs the Core Control to let the downlink packets to flow, providing the tunnel endpoint to which the packets have to be sent. Then, the Core Control instructs the S-GW using the tunnel endpoint just received.

The efficiency improvements brought by the proposed procedure are twofold: on one hand, we avoid the network load caused by the previous data loop, and on the other hand, the procedure requires less signals than the original S1 handover procedure. The whole procedure is extensively described in [107].

5.5 Signaling Load Analysis

In this section, we propose the same study we have already performed during the previous refactoring step. We start providing the equations that express the costs of handling each one of the network events considered. Please notice how the equations change when compared to the ones of the first refactoring step: the procedures in the new model require a different set of interfaces, which are used with different frequencies. In the last part, we summarize the coefficients of the equations and we compare the new model with the current LTE architecture.

5.5.1 Interfaces

The list of variables follows:

- $c1$ corresponds to the cost associated to interface LTE-Uu, between UE and eNB Data;
- $c2$ corresponds to the cost associated to interface S1-U, between eNB Data and S-GW Data;
- $c3$ corresponds to the cost associated to interface S5, between S-GW Data and P-GW Data;
- $c4$ corresponds to the cost associated to interface S1-MME, between eNB Data and RAN Control;
- $c5$ corresponds to the cost associated to interface S11, between S-GW Data and Core Control;
- $c6$ corresponds to the cost associated to interface Gx, between P-GW Data and Core Control;
- $c7$ corresponds to the cost associated to interface S6a, between Ran Control and HSS/SPR;
- $c9$ corresponds to the cost associated to interface X2, between two eNB Data parts (if the eNBs are linked);
- $c15$ corresponds to the cost associated to interface RC-CC, between RAN Control and Core Control.

Please notice that the interface Sp is no more used, since the direct link between Core Control and SPR is now missing: the interface S6a is used in place of it.

5.5.2 Initial Attach

The list of messages exchanged during initial attach follows:

- 13 messages exchanged over LTE-Uu interface (Steps 1-2-3-4-5-10-11-14-15-24-25-26-27);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 7 messages exchanged over S1-MME interface (Steps 6-9-12-13-16-23-28);

- 2 messages exchanged over S11 interface (Steps 18-19);
- 2 messages exchanged over Gx interface (Steps 20-21);
- 2 messages exchanged over S6a interface (Steps 7-8);
- 0 messages exchanged over X2 interface;
- 2 messages exchanged over RC-CC interface (Steps 17-22).

The total amount of messages required by the procedure is 28. The total amount of time requested by the message exchanging in the procedure may be expressed as:

$$tot\_time = 13 * c1 + 7 * c4 + 2 * c5 + 2 * c6 + 2 * c7 + 2 * c15$$

5.5.3 Active_to_Idle Transition

The list of messages exchanged during initial attach follows:

- 1 message exchanged over LTE-Uu interface (Steps 7);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 3 messages exchanged over S1-MME interface (Steps 1-6-8);
- 2 messages exchanged over S11 interface (Steps 3-4);
- 0 messages exchanged over Gx interface;
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over X2 interface;
- 2 messages exchanged over RC-CC interface (Steps 2-5).

The total amount of messages required by the procedure is 8. The total amount of time requested by the message exchanging in the procedure may be expressed as:

$$tot\_time = c1 + 3 * c4 + 2 * c5 + 2 * c15$$

5.5.4 Idle_to_Active Transition

5.5.4.1 UE Triggered

The list of messages exchanged during initial attach follows:

- 8 messages exchanged over LTE-Uu interface (Steps 1-2-3-4-5-8-9-10);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 3 messages exchanged over S1-MME interface (Steps 6-7-11);
- 2 messages exchanged over S11 interface (Steps 13-14);
- 0 messages exchanged over Gx interface;

- 0 messages exchanged over S6a interface;
- 0 messages exchanged over X2 interface;
- 2 messages exchanged over RC-CC interface (Steps 12-15).

The total amount of messages required by the procedure is 15. The total amount of time requested by the message exchanging in the procedure may be expressed as:

$$tot\_time = 8 * c1 + 3 * c4 + 2 * c5 + 2 * c15$$

5.5.4.2 Network Triggered

The list of messages exchanged during initial attach follows:

- 9 messages exchanged over LTE-Uu interface (Steps 4-5-6-7-8-9-12-13-14);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 4 messages exchanged over S1-MME interface (Steps 3-10-11-15);
- 3 messages exchanged over S11 interface (Steps 1-17-18);
- 0 messages exchanged over Gx interface;
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over X2 interface;
- 3 messages exchanged over RC-CC interface (Steps 2-16-19).

The total amount of messages required by the procedure is 19. The total amount of time requested by the message exchanging in the procedure may be expressed as:

$$tot\_time = 9 * c1 + 4 * c4 + 3 * c5 + 3 * c15$$

5.5.5 Handover

5.5.5.1 X2 Handover

The list of messages exchanged during initial attach follows:

- 3 messages exchanged over LTE-Uu interface (Steps 1-6-9);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 9 messages exchanged over S1-MME interface (Steps 2-3-4-5-7-8-10-15-16);
- 2 messages exchanged over S11 interface (Steps 12-13);
- 0 messages exchanged over Gx interface;
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over X2 interface;

- 2 messages exchanged over RC-CC interface (Steps 11-14).

The total amount of messages required by the procedure is 16. The total amount of time requested by the message exchanging in the procedure may be expressed as:

$$tot\_time = 3 * c1 + 9 * c4 + 2 * c5 + 2 * c15$$

5.5.5.2 S1 Handover

The list of messages exchanged during initial attach follows:

- 3 messages exchanged over LTE-Uu interface (Steps 1-10-13);
- 0 messages exchanged over S1-U interface;
- 0 messages exchanged over S5 interface;
- 9 messages exchanged over S1-MME interface (Steps 2-7-8-9-11-12-14-19-20);
- 4 messages exchanged over S11 interface (Steps 4-5-16-17);
- 0 messages exchanged over Gx interface;
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over X2 interface;
- 4 messages exchanged over RC-CC interface (Steps 3-6-15-18).

The total amount of messages required by the procedure is 20. The total amount of time requested by the message exchanging in the procedure may be expressed as:

$$tot\_time = 3 * c1 + 9 * c4 + 4 * c5 + 4 * c15$$

5.5.6 Summary

Table 5.1 summarizes the results from the previous sections.

	IA	**AtI**	**ItA (UE)**	**ItA (Net)**	**X2H**	**S1H**	**Total**
LTE-Uu	13	1	8	9	3	3	37
S1-U	0	0	0	0	0	0	0
S5	0	0	0	0	0	0	0
S1-MME	7	3	3	4	9	9	35
S11	2	2	2	3	2	4	15
Gx	2	0	0	0	0	0	2
S6a	2	0	0	0	0	0	2
Sp	0	0	0	0	0	0	0
X2	0	0	0	0	0	0	0
RC-CC	2	2	2	3	2	4	15
Total	28	8	15	19	16	20	

Table 5.1: **Signaling load analysis of the second proposed model.** *The compression of the control plane entities is beneficial, introducing a limited amount of signals on the only new interface (RC-CC), and bringing a significant reduction in the signaling load at the same time.*

Instead, Table 5.2 expresses the differences on the signaling load between the proposed model and the current LTE architecture.

	IA	AtI	ItA (UE)	ItA (Net)	X2H	S1H	Total
LTE-Uu	0	0	0	0	0	0	0
S1-U	0	0	0	0	0	0	0
S5	-2	0	0	0	-2	-2	-6
S1-MME	-1	0	0	0	+7	0	+6
S11	-2	0	0	-1	0	-2	-5
Gx	0	0	0	0	-2	-2	-4
S6a	-2	0	0	0	0	0	-2
Sp	-2	0	0	0	0	0	-2
X2	0	0	0	0	-4	0	-4
RC-CC	+2	+2	+2	+3	+2	+4	+15
Total	-7	+2	+2	+2	+1	-2	

Table 5.2: **Comparison between the current LTE architecture and the second proposed model.** *The saving of signals in initial attach and S1 handover, together with the small overhead in the remaining events, make the proposed model a reasonable alternative to the current LTE architecture.*

The proposed model is a good compromise between the benefits of software-defined networking and its drawbacks. In fact, the resulting architecture has still a good separation between control and data planes, and the overheads introduced in handling the network events are limited. The procedure for the initial attach requires now 7 signals less that the traditional procedure for the same event. Also the S1 handover procedure asks for two signals less than the original counterpart, and the remaining procedures always ask for less than 3 additional signals. Please notice also the difference in the usages of the interfaces: the proposed model requires less signals on all the original interfaces, except for S1-MME. This is because part of the signals are offloaded on the introduced RC-CC interface, and therefore this could seem a fake benefit. Nevertheless, we are requiring just one additional interface in this model, compared to the five we needed in the previous refactoring step.

5.6 The Network Functions

Network Element	**Tasks**
RAN Control	[S01], [S04], [S05], [S06], [S07], [S08], [S09], [S10], [S11], [S12], [S13], [S14], [S15], [S17], [S20], [S21], [S22], [S23], [S24]
eNB Data	[S03], [S25], [M30]
Core Control	[S02], [S16], [S18], [S19]
S-GW Data	[S28], [S29], [M30], [S31]
P-GW Data	[M30]
HSS / SPR	No tasks

Table 5.3: **Tasks assignment after the second refactoring step.** *The elements of the forwarding layer take care only of QoS rules enforcement and tasks linked with their physical nature, as handling of radio identifiers, or acting as the base stations' anchor point.*

The task assignment in Table 5.3 reflects the architectural choices performed during the refactoring step. The RAN Control executes most of the tasks, as it corresponds to the union of the old eNB Control, MME, and PCRF, and therefore the tasks which were assigned to them are naturally coalesced in this entity. The Core Control takes care about the tunnel identifiers related with S-GW and P-GW, and moreover handles the IP addresses assignment. Taking a look to the elements of the forwarding layer, their main aim is the enforcement of the QoS rules received from the upper layer. The remaining tasks that are assigned to the data elements are linked with their specific physical nature: as an example, the eNB Data handles the C-RNTI assignment, since it is equipped with the radio interface. Please notice that the buffering of downlink packets during handover is now shifted from the eNB to the S-GW, since we have designed the corresponding procedure to avoid the data loop between SeNB and S-GW.

Chapter 6

Refactoring Step 3: Shifting the Functionalities to the RAN

From the last refactoring step we have gained a software-defined cellular architecture which introduces small overheads in the handling of most of the considered network events. Even if the result is appreciable, we still have not reached our original aim, which is the reduction of the impact of signaling storms. Therefore, we performed a last refactoring step to further reduce the number of signals in the procedures triggered by the network events, but keeping the software-defined interface. In this chapter we describe the ideas that we have pursued in order to reach this twofold aim.

6.1 The Brain of the Network in the RAN

In the previous refactoring step, we have compressed the entities of the control plane in order to gain just two network elements, the RAN Control and the Core Control. We have thought about them as two distinct remote servers, which are linked to eNB, S-GW, P-GW, and potentially to some intermediate gateways as well. We already know that RAN Control and Core Control are actually merely-software entities, and, from the signaling load analysis of the last proposed model, we know also that RC-CC and S1-MME are two heavily-used interfaces. These two considerations made us thinking about merging together the software entities with the hardware ones, which is, putting the control entities inside (or close to) the forwarding ones.

It could seem that this idea is absolutely insane, as we have worked till now in order to gain a clear separation between control and data planes. We argue that this merging is possible, without affecting the designed software-defined interface, and gaining the obvious advantages of compressing entities. In fact, the work performed in the previous refactoring step has allowed to gain a set of basic building blocks, which corresponds to the set of entities we have identified. Actually, there is no need of mapping each single building block to a different physical entity, but we can rather coalesce more than one building block on the same device, depending on the device itself.

Suppose to have a device equipped with a radio interface and a huge computational capacity at the same time. According to our previous understanding, we have to decide

if we are going to use it as eNB Data or if we want it to be our RAN Control. What we argue now is that we can put both entities in the same device instead, thanks to the modularization work performed during the previous steps. In fact, the RAN Control is a software-only entity, and can be freely put inside the eNB, or in a dedicate server, as the interface between it and its controlled entity is clearly defined.

Given this context, we argue that most of the functionalities (*i.e.* both software and hardware requirements) should be put as close to the RAN as possible. In fact, many works report the wastage of processing capacity in the base stations, which can go up to 80% [86, 125]. In this scenario, it may be beneficial to leverage this free computational power carrying out the control plane tasks inside the eNBs. Moreover, An *et al.* [30] promote the execution of the control plane as close as possible to the network's customers, in order to reduce the latencies of the signals processing. Nevertheless, the resulting efficiency is not the only driving motivation. The work of Thompson *et al.* [122] encourages to push the network services to the edge in order to gain almost-autonomous base stations, with minimal support from a thin core network. In fact, the fault tolerance of a system decreases with the number of its components, and therefore a network composed by few elements it is more likely to survive during earthquakes, tsunamis, and other natural disasters. In this context, having most of the network functionalities inside the eNBs could be crucial in the survival of the original cellular network, and therefore in the organization of the rescue operations.

Finally, we have considered the roles of S-GW, P-GW, and any intermediate gateway in the model coming out from the last refactoring step. As we know, we have lifted their software-requirements up to the control layer, keeping in them just the hardware requirements. The S-GW has the hardware property of being the first junction point between the eNBs, while the P-GW has the hardware property of being the last gateway before exiting the cellular network domain. Similarly to the previous situation, we realized that these properties may be mapped to the same physical device, as there is no need of keeping them in two distinct ones. If we have a single gateway with buffering capabilities, we can use it both as junction point between the eNBs, and as last gateway before reaching the Internet.

Therefore, the ideas behind the last refactoring step can be summarized as follows: shifting to the RAN as much of the functionalities as possible, and reduce the number of physical entities depending on the availabilities. We propose a modular architecture which can be mapped to an arbitrary number of physical devices, and this is exactly the kind of flexibility we were looking for.

6.2 The Refactoring Principles

Since we start the third refactoring step from the output of the second one, we are already embodying all the refactoring principles of the second refactoring step. We further refine the list with the following principles:

- *Shift as much functionalities as possible to the RAN Control.* This means that the Core Control executes only tasks that have to be executed in the Core Control mandatorily. All the other tasks are executed by the RAN Control.

- *Abstract the physical requirements from the need of separated physical devices.* Instead of declaring that we need a S-GW and a P-GW, we want to say what we need of a S-GW and what we need of a P-GW. In this context, we are able to

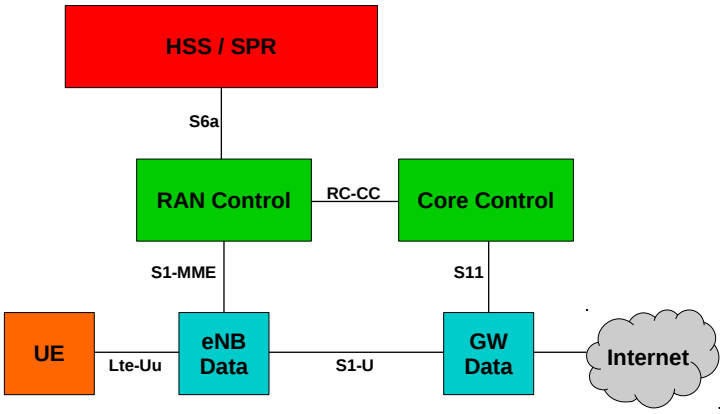


Figure 6.1: **The model after the third refactoring step.** *The physical properties of S-GW and P-GW have been coalesced in a single generic gateway.*

map these characteristics to an arbitrary number of physical devices, depending on our availabilities.

- *The state variables used by a single data plane element are handled by its corresponding control plane element only.* As we noticed some cases of useless redundancy of state variables among the control plane entities, we adopt this rule to ensure that information used locally are stored locally as well.
- *Ensure that every message from a control plane element is acknowledged.* As we do not want to make any assumption on the reliability of our interfaces, we want every control plane message to be acknowledged.

The rationale behind the last refactoring principle is that we are able to depict the worst case in this way. This is particularly useful because we can further reduce the signals needed removing the acknowledgement in case we have some reliable interfaces. Again, the outcome of this refactoring step is a flexible architecture, which can be tuned depending on our availabilities.

Please notice that the acknowledgements are possibly sent in piggyback with other messages, in order to avoid the need of an explicit acknowledgement signal as much as possible. In order to ensure the presence of an acknowledgement for each control plane message, the idea of chain of messages has been extensively used. What is a chain of messages and which are its properties are explained in Appendix D.

6.3 The Proposed Model

Figure 6.1 represents a schematic view of the architecture resulting after the last refactoring step. We immediately notice that it is very similar to the previous one, since the only difference consists in the substitution of S-GW and P-GW with a generic GW device. This generic GW actually matches a gateway which is acting

both as mobility anchor and as entry and exit point from the cellular network. We have therefore merged the requirements of S-GW and P-GW in a single physical device, as in both cases we have no specific hardware requirements, but rather we have requirements about the position of the device in the network. We have designed the generic GW to meet the requirements coming from S-GW and P-GW at the same time. This does not bring only a smaller number of signals, as the device to configured is one instead of two, but it also implies a smaller number of state variables to handle, as the tunnel identifiers for the bearer linking S-GW and P-GW are no more needed.

The key point of the last refactoring step is the following one: the mapping between the entities of the architecture and the physical devices is arbitrary. This means that a one-to-one mapping, and therefore having a different physical device for each entity, represents only one possible scenario, and it also represents the worst-case one. If we have a hardware device with high computational capabilities and a radio interface, then we can put both RAN Control and eNB Data in it, having the benefit that the cost associated to the S1-MME interface drops to zero. Similarly, if we have a gateway with computational capabilities, we can put the Core Control together with the GW Data in it. We can also act in the opposite way, in case we have a reliable interface that we would like to leverage in our architecture. Again, we underline that this is possible only because of the clear separation between the duties performed in the previous refactoring steps: once the roles are well defined, they become composable.

6.4 The Network Events

In the following, we describe the main characteristics of the procedures triggered by the considered network events, in the same way as we have done in the previous refactoring steps. The reader can find the details of all the modified procedures in [115].

6.4.1 Initial Attach

The key differences between the procedure for the proposed model and the original one are the same of the previous refactoring step. We have a single access to the storage layer, in which we retrieve all the information needed during the procedure. The RAN Control generates almost all the state variables, except for the tunnel identifiers of the GW and the IP of the UE. Since we have a single device to instruct (*i.e.* GW) instead of two (*i.e.* S-GW and P-GW), the procedure requires two signals less than the one of the previous modelling step.

There are some remarkable differences in the handling of the state variables. Here we point out the most important ones:

- There is no need of handling S5 TEID UL and S5 TEID DL, as they are the tunnel identifiers of the data bearer which was linking S-GW and P-GW. The work of the Core Control is therefore further simplified.
- During the original initial attach procedure, the MME resolves the IP address of the P-GW to reach, and it provides the S-GW with this information, such that it can direct the data packets accordingly. In the new model, the P-GW is represented by the generic GW gateway, and therefore the RAN Control provides the IP of the gateway to the eNB instead. Having coalesced the S-GW with the P-GW, the function is simply executed in the previous part of the whole data tunnel.

- The QoS values provided to the generic GW are the ones which are given to the P-GW in the original LTE procedure, as they consist in a super-set of the ones given to the S-GW.

- To the best of our knowledge, ECGI and TAI are used only for the generation of the QoS rules. Therefore, these state variables are not spread in the whole architecture, but they are rather kept just in the RAN Control and in the eNB Data.

- The IMSI is used to retrieve customer-specific values from the storage layer and to associate an IP address to the device. Therefore, the only two entities which are aware of this state variable are the RAN Control and the Core Control.

- The information used temporarily are stored temporarily as well, meaning that they are discarded after their usage. This happens with the QoS values retrieved from HSS and SPR, and also with the Default APN variable.

In general, the procedure of the last model greatly reduces the signaling load and provides a smarter handling of the state variables as well.

The whole procedure is extensively described in [114].

6.4.2 Active_to_Idle Transition

The situation is the same depicted by the previous refactoring step, as the new model does not bring explicit improvements for this network event. Instead of notifying the S-GW, the Core Control notifies the generic GW, but there are no signals saved. Also from the state variables perspective there are no changes in the procedure. Therefore, the scenario is the same depicted by the previous refactoring step: the procedure still requires two more signals when compared to the original one.

The whole procedure is extensively described in [110].

6.4.3 Idle_to_Active Transition

In this section we describe the differences in the procedure for transiting from Idle to Active state between the one of the new model and the one of the original LTE architecture. Similarly to the previous refactoring steps, we perform the comparison on the two triggering sources separately.

6.4.3.1 UE Triggered

The procedure of the last proposed model does not decrease the number of signals needed by the model coming out from the second refactoring step. In fact, the only entity which has changed is the generic GW, which substitutes the S-GW, but nevertheless the indirection between the RAN Control and the GW is still present, and therefore the procedure still requires two signals more than its corresponding one in the original LTE architecture. There are no significant improvements neither on the handling of state variables.

The whole procedure is extensively described in [113].

6.4.3.2 Network Triggered

Similarly to the previous case, the situation is the same depicted in the comparison with the model after the second refactoring step. In fact, the procedure requires the usual four messages more in order to notify the UE to start the transition from Idle to Active state. As the original LTE procedure requires the four signals as well, the output of the comparison is exactly the same: the procedure of the new model requires two signals more. Since the four additional signals do not carry any state variable, the procedure does not change the handling of the state variables.

The whole procedure is extensively described in [112].

6.4.4 Handover

We have designed a single procedure for handling the handover, regardless the presence of a direct link between the two considered eNBs (*i.e.* a X2 interface). Indeed, we argue that the eNBs are data plane elements, and therefore any communication channel between them should be used for data communications only. Since the procedure for handling handover consists of control plane messages only, there is no point in using the X2 interface (if any).

Moreover, the X2 interface is used as part of the indirect data tunnel, which brings the downlink packets from the S-GW to the SeNB ,and from the SeNB to the TeNB, during the handover procedure. As already pointed out in the previous refactoring step, we believe that this indirection is completely useless, as the downlink packets could be buffered directly at the gateway, instead of buffering them at the TeNB. Therefore, we have designed a single procedure, which is agnostic about the presence of any X2 interface between SeNB and TeNB.

The idea behind the procedure is straightforward: when the handover is triggered, the Core control instructs the GW to start buffering the downlink packets, then when the link between the UE and the TeNB is operational, the Core Control instructs the GW to stop buffering, and to let the buffered packets to flow to the UE. Please notice that this does not hold for the uplink packets: every time the UE has an uplink packet to send, it can send it to the Internet regardless if through the SeNB or through the TeNB, as the path the packet has to follow is exactly the same.

The comparisons with the original LTE procedures give different results. The designed procedure requires five signals more than the traditional X2 handover procedure, while it asks for two signals less when compared to the original S1 handover procedure. Again, we are in a situation in which the benefits of the separation between control and data plane contrast with its efficiency drawbacks. Nevertheless please notice that we have some different efficiency benefits, thanks to the buffering of the downlink packets at the GW. Actually, the benefit is twofold: from a theoretical point of view, it is absolutely correct to use the link between SeNB and GW only for packets from and to the UEs linked to the SeNB; from a practical point of view, the link between SeNB and GW is less congested in this way.

The whole procedure is extensively described in [111].

6.5 Signaling Load Analysis

We provide here the equations used to estimate the costs in adopting the proposed model, and then we compare the model with the current LTE architecture on the number of signals required. Please notice that the interfaces used by the proposed

model are the same of the model coming out from the second refactoring step, except for the Gx interface, since it was linking P-GW to the Core Control, and therefore it is not required anymore.

6.5.1 Interfaces

The list of variables follows:

- $c1$ corresponds to the cost associated to interface LTE-Uu, between UE and eNB Data;
- $c2$ corresponds to the cost associated to interface S1-U, between eNB Data and GW;
- $c4$ corresponds to the cost associated to interface S1-MME, between eNB Data and RAN Control;
- $c5$ corresponds to the cost associated to interface S11, between GW and Core Control;
- $c7$ corresponds to the cost associated to interface S6a, between RAN Control and HSS/SPR;
- $c9$ corresponds to the cost associated to the interface X2, between two eNB Data parts (if the eNBs are linked);
- $c15$ corresponds to the cost associated to the interface RC-CC, between RAN Control and Core Control.

6.5.2 Initial Attach

The list of messages exchanged during Initial Attach follows:

- 13 messages exchanged over LTE-Uu interface (Steps 1-2-3-4-5-10-11-14-15-22-23-24-25);
- 0 messages exchanged over S1-U interface;
- 7 messages exchanged over S1-MME interface (Steps 6-9-12-13-16-21-26);
- 2 messages exchanged over S11 interface (Steps 18-19);
- 2 messages exchanged over S6a interface (Steps 7-8);
- 0 messages exchanged over X2 interface;
- 2 messages exchanged over RC-CC interface (Steps 17-20).

The total amount of messages required by the procedure is 26. The total amount of time requested by the message exchanging in the procedure can be expressed as:

$$tot\_time = 13 * c1 + 7 * c4 + 2 * c5 + 2 * c7 + 2 * c15$$

6.5.3 Active_to_Idle Transition

The list of messages exchanged during Active_to_Idle Transition follows:

- 1 message exchanged over LTE-Uu interface (Steps 7);
- 0 messages exchanged over S1-U interface;
- 3 messages exchanged over S1-MME interface (Steps 1-6-8);
- 2 messages exchanged over S11 interface (Steps 3-4);
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over X2 interface;
- 2 messages exchanged over RC-CC interface (Steps 2-5).

The total amount of messages required by the procedure is 8. The total amount of time requested by the message exchanging in the procedure can be expressed as:

$$tot\_time = c1 + 3 * c4 + 2 * c5 + 2 * c15$$

6.5.4 Idle_to_Active Transition

6.5.4.1 UE Triggered

The list of messages exchanged during Idle_to_Active Transition - UE Triggered follows:

- 8 messages exchanged over LTE-Uu interface (Steps 1-2-3-4-5-8-9-10);
- 0 messages exchanged over S1-U interface;
- 3 messages exchanged over S1-MME interface (Steps 6-7-11);
- 2 messages exchanged over S11 interface (Steps 13-14);
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over X2 interface;
- 2 messages exchanged over RC-CC interface (Steps 12-15).

The total amount of messages required by the procedure is 15. The total amount of time requested by the message exchanging in the procedure can be expressed as:

$$tot\_time = 8 * c1 + 3 * c4 + 2 * c5 + 2 * c15$$

6.5.4.2 Network Triggered

The list of messages exchanged during Idle_to_Active Transition - Network Triggered follows:

- 9 messages exchanged over LTE-Uu interface (Steps 4-5-6-7-8-9-12-13-14);
- 0 messages exchanged over S1-U interface;
- 4 messages exchanged over S1-MME interface (Steps 3-10-11-15);

- 3 messages exchanged over S11 interface (Steps 1-17-18);
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over X2 interface;
- 3 messages exchanged over RC-CC interface (Steps 2-16-19).

The total amount of messages required by the procedure is 19. The total amount of time requested by the message exchanging in the procedure can be expressed as:

$$tot\_time = 9 * c1 + 4 * c4 + 3 * c5 + 3 * c15$$

6.5.5 Handover

The list of messages exchanged during Handover follows:

- 3 messages exchanged over LTE-Uu interface (Steps 1-10-13);
- 0 messages exchanged over S1-U interface;
- 9 messages exchanged over S1-MME interface (Steps 2-7-8-9-11-12-14-19-20);
- 4 messages exchanged over S11 interface (Steps 4-5-16-17);
- 0 messages exchanged over S6a interface;
- 0 messages exchanged over X2 interface;
- 4 messages exchanged over RC-CC interface (Steps 3-6-15-18).

The total amount of messages required by the procedure is 20. The total amount of time requested by the message exchanging in the procedure can be expressed as:

$$tot\_time = 3 * c1 + 9 * c4 + 4 * c5 + 4 * c15$$

6.5.6 Summary

Table 6.1 summarizes the results from the previous sections.

	IA	**AtI**	**ItA (UE)**	**ItA (Net)**	**Hand**	**Total**
LTE-Uu	13	1	8	9	3	34 (+3)
S1-U	0	0	0	0	0	0
S1-MME	7	3	3	4	9	26 (+9)
S11	2	2	2	3	4	13 (+4)
S6a	2	0	0	0	0	2
X2	0	0	0	0	0	0
RC-CC	2	2	2	3	4	13 (+4)
Total	26	8	15	19	20	

Table 6.1: **Signaling load analysis of the third proposed model.** *As there is a single handover procedure, its cost has to be added two times in order to get comparable total values.*

Table 6.2 presents the comparison with the current LTE architecture instead.

	IA	AtI	ItA (UE)	ItA (Net)	X2H	S1H	Total
LTE-Uu	0	0	0	0	0	0	0
S1-U	0	0	0	0	0	0	0
S5	-2	0	0	0	-2	-2	-6
S1-MME	-1	0	0	0	+7	0	+6
S11	-2	0	0	-1	+2	-2	-3
Gx	-2	0	0	0	-2	-2	-6
S6a	-2	0	0	0	0	0	-2
Sp	-2	0	0	0	0	0	-2
X2	0	0	0	0	-4	0	-4
RC-CC	+2	+2	+2	+3	+4	+4	+17
Total	-9	+2	+2	+2	+5	-2	

Table 6.2: **Comparison between the current LTE architecture and the third proposed model.** *The total number of signals saved equals to the total number of signals added by the model.*

The result of the comparison is similar to the one of the previous refactoring step. The further benefits brought on the initial attach and S1 handover handling are compensated by additional signals on all the remaining procedures. As already pointed out previously, the number of signals is just one of the many perspectives with which we can examine the architecture. In fact, we know that behind the overhead on the X2 handover we actually have another kind of efficiency.

Nevertheless, till now it seems that the three refactoring steps brought architectural advantages and some efficiency benefits (*e.g.* state variables handling, no indirect tunnels), but they did not bring a complete solution to the original problem, which are signaling storms. In fact, the total signals required by the final model is exactly the same of the original LTE architecture. This means that we have essentially shifted the burden among the procedures, lightening some of them, and loading the remaining ones. Section 6.7 shows which is the true outcome of the refactoring steps, and how this can be leveraged in order to solve signaling storm as well.

6.6 The Network Functions

Network Element	Tasks
RAN Control	[S01], [S04], [S05], [S06], [S07], [S08], [S09], [S10], [S11], [S12], [S13], [S14], [S15], [S17], [S20], [S21], [S22], [S23], [S24]
eNB Data	[S03], [S25], [M30]
Core Control	[S02], [S16]
GW Data	[S28], [S29], [M30], [S31]
HSS / SPR	No tasks

Table 6.3: **Tasks assignment after the third refactoring step.** *The GW Data has the tasks of the old S-GW, and tasks [S17] and [S18] are no more needed thanks to the merging of S-GW and P-GW in the same GW entity.*

Table 6.3 shows how the tasks are assigned to the entities of the proposed model. There are no substantial differences with the previous refactoring step, as the entities involved are essentially the same. The tasks assigned to the GW are the ones that were assigned to the S-GW, and this happens because of two reasons: the first is that the GW actually acts as the S-GW of the architecture, and the second one is that the tasks of the S-GW consist in a super-set of the tasks of the P-GW, and therefore assigning the tasks of the S-GW to the GW implicitly makes the GW acting as the P-GW as well.

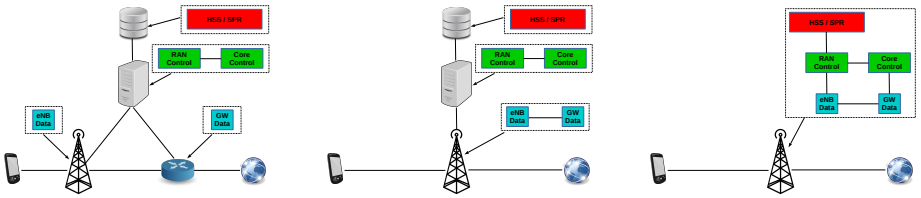
Please notice that tasks [S17] and [S18] are not associated with any entity. These tasks correspond to the handling of the tunnel identifiers for the data bearer between S-GW and P-GW. Since these two entities have been merged in the same entity, the generic GW, these tunnel identifiers do not need to be handled by the entities of the architecture anymore. Less state variables imply lighter network state, which is beneficial for the network on several aspects: signals with less information, lighter memory requirements, smaller recovery time for the network after a fault, and so on.

Looking at the tasks assigned to the Core Control, we can notice that it still takes care about the IP assignment. This could seem illogical, as the main idea behind this refactoring step is to shift as much functionalities as possible to the RAN Control, and the IP assignment seems a functionality suitable for this purpose. Nevertheless, we have to remember that the GW is the last gateway before reaching the Internet, and essentially it separates the cellular network from the rest of the network. Therefore, the GW is the place in which any NAT service is executed, and the GW has also to know to which eNB each downlink packet has to be forwarded. If the knowledge about the IP assignment was shifted to the RAN, it would require a tight communication between the control entities, because the GW always needs these IP information in order to forward the traffic accordingly. Under this perspective, it is better to keep this functionality in the Core Control: in the same way in which in a domestic WLAN the wireless gateway, being the last point before the Internet, performs IP assignment and executes NAT services, the GW takes care about the same tasks in a cellular network.

6.7 An Unbridled Flexibility

In this section we want to explain what we believe to be the real outcome of the three refactoring steps that have been performed. We have started our refactoring process from a convoluted architecture, in which control and data planes were mixed, and in which was difficult to understand if some devices could be actually merged together, or split in smaller parts. We have focused on creating a clear separation of the duties in the architecture, and then we have associated the entities to the appropriate category of duties. Finally, we have collapsed together some entities belonging to the same layer.

The resulting architecture is a set of composable building blocks, in the sense that they can be easily put together and split, depending on the availabilities of the network manager. In fact, each building block consists in requirements that may be mapped to a suitable hardware device. The storage layer expresses the necessity of a database-like structure, no more than that. Both control entities, being essentially software entities, simply require a hardware device with computational capabilities. The forwarding layer has three precise requirements: a radio interface, a device acting as junction point for possibly many radio interfaces, and a device acting as entry or



(a) Control entities collapsed in the same physical device.

(b) Control and forwarding entities collapsed in two different physical devices.

(c) All the entities collapsed in the same physical device.

Figure 6.2: **Possible implementations of the proposed model.** *The more the entities are collapsed, the higher are the requirements on the physical devices, but the lighter is the signaling load.*

exit point of the cellular network[1]. Once we have this list of requirements, it is up to the network manager to map them to his owned devices.

Figure 6.2 shows some possible implementations of the final model. In particular, Figure 6.2a depicts an implementation whose requirements are extremely constrained. In fact, the only optimization we are introducing is the merging of RAN Control and Core Control in the same physical device.

Here, we actually need a base station, a gateway, a remote server, and a remote database, and no modifications to the model are needed, as it is enough to put the two software entities (*i.e.* the control ones) in the same execution environment.

The benefit of this merging is that the cost associated to the RC-CC interface immediately drops to zero. This implies that we are going to experience a huge saving in the number of signals required by the procedures of the last model. This happens just because we have decided to avoid the usage of an additional server, which would have been required to put the Core Control in a physically separated device. Please notice that the merging the management of RAN and core network in the same entity is not a completely new idea, as it has been already studied in order to reduce the latencies coming from the signaling processing [60].

Figure 6.2b shows a scenario in which the network manager has a base station with gateway capabilities. This essentially means that we can map all the forwarding layer requirements on the same physical device. Please notice that the other eNBs would require to be directly linked with the eNB with gateway capabilities in this case, as that one is acting as mobility anchor as well. Figure 6.2c shows the optimal scenario, in which we have a physical device with all the feature that we actually need from our cellular network: in this context, the signal saving is maximum, as all the signals would be internal to the same device.

Therefore, the real outcome of the refactoring process is the flexibility that has been injected in the cellular architecture. The network manager can leverage the reliable interfaces he has, and at the same time he can compress the entities whose linking interfaces are particularly weak. In this context, the solution of the signaling storm comes as a nice side effect.

[1] An implicit requirement of the forwarding layer is that the devices have to allow the enforcement of QoS rules.

Chapter 7

Analysis and Results

After the refactoring process, we proceeded analyzing deeply the proposed model, as the preliminary analysis performed in the previous sections is not enough to judge the goodness of the work outcome. In fact, the initial analysis provides us just the number of times each interface is used for handling each event. We can therefore compare the proposed model and the original LTE architecture just on single occurrences of network events, as we have done in our previous analysis. What we need to perform a more complete analysis is a list of frequencies for each network event, *i.e.* we need to know how much frequently each network event occurs, such that we can compute the total number of signals saved or additionally required.

In order to proceed with this kind of analysis, there are two suitable approaches. The first one consists in implementing the proposed model in a LTE simulator. Once we have the implementations of the current LTE architecture and the proposed model, we can use the input frequencies of network events we prefer, picking them from data collected from real LTE networks for example, and we can compare the two architectures. In this scenario, the implementation acts as a proof of concept, in the sense that we show that our proposed model is actually feasible.

The second one is a more analytical approach, in which we simply use the input frequencies together with the factors from the primary analysis in order to gain directly the final values for the comparison. Therefore we skip the implementation phase in this case, even if the output of the two approaches is exactly the same.

Initially we have chosen to follow the first approach. Nevertheless, we faced great troubles during the implementation process because of different motivations, and therefore we have changed the approach eventually. The following sections describe the simulators we have tried, the issues we have faced with them, and the analysis process performed following the second approach.

7.1 LTE Simulators

As the first intention was to follow the first approach, we have examined a long list of simulators, looking for the one suitable for our aims. In the following, we describe briefly each simulator, pointing out pro and cons, and explaining why it has non been chosen as our implementation basis.

Before going through each simulator, we want to express precisely which were our requirements for it. This helps in understanding the decision to turn in the other analysis approach eventually. We were looking for a simulator which:

- is open-source (or the source code is available for academic purposes), such we are able to implement our proposed model modifying the implementation of the current LTE architecture;
- provides the documentation for users, as we need to test the current LTE architecture as well;
- provides the documentation for developers, such that we can modify the implementation of the current LTE architecture to model the proposed architecture, and we can test it subsequently. The requirement aims to guarantee that the developers have designed the simulator with the explicit aim of further developments, such that the understanding of the internal code should result in a straightforward activity;
- is possibly free-of-charge, in order to avoid funding requests for licenses;
- is possibly software-only, such that we do not need any specific hardware for the execution of the experiments;
- is a system-level simulator, since we are not interested in PHY/MAC layer features (inter-cell interference management, scheduling of user jobs, etc.) but we are more interested in message exchanges among the elements of E-UTRAN and EPC;
- represents as much EPC entities as possible. As we are interested in every single message exchanged during the handling of network events, we would like to avoid fusion of different entities in a single device (*e.g.* S-GW and P-GW together);
- represents the events in which we are interested (initial attach, transition Idle-Active, transition Active-Idle, and handover).

The next sections describe how none of the examined simulators is able to meet all these requirements. We want this list to act as a call for simulators, as a software tool with all these features would boost the research work in this area.

7.1.1 LTE-A Downlink System Level Simulator

The tool is a software that replicates accurately the physical level details of the interaction between UE and eNB. Unfortunately, the tool is not suitable for several reasons.

First, even if the simulator is a system level one, it is extremely focused on the link level details, as the physical properties of the radio channel. Instead, we are interested in the information carried by the signals, rather in their intrinsic characteristics.

Second, the simulator is focused on the interaction between UE and eNB. We are rather interested in the signals going through all the components of the cellular network, and not only in its final part.

Third, the simulator reproduces downlink communications only. We are interested in signals going both ways instead.

Finally, there is no documentation related to further developments of the tool. This makes really hard the modification of the tool in order to represent the proposed model. We are rather interested in a tool designed with the idea of being modified: this guarantees flexibility and easiness in implementing ideas.

The tool is presented in the work of Ikuno *et al.* [68], and it is available at [14].

7.1.2 SimuLTE

The tool is an extension of the OMNet++ framework, which is a multi-purposes network simulator. This condition leads to several advantages: stability, reliability, easiness in further developments, easiness in usage, and support by a large community of users.

Nevertheless, there are three main disadvantages in using this tool. The first one is that the simulator implements only the user plane of the cellular network. Instead, we are mostly interested in the control plane messages, which consist in the signals exchanged during the considered network events.

The second one is that the simulator represents only UE and eNB, similarly to the previous tool considered. As already explained, this is a huge limitation in our analysis scenario.

Finally, the simulator does not represent the handover events. This implies that we would be forced to implement the handling of handover events by ourself. As the tool deals with details which are not in the scope of our study (*e.g.* lower level details), we would be forced to implement them as well, which consists in a concrete risk of injecting errors in the software.

For these motivations, the tool has been excluded by our list. The tool is presented in the work of Virdis *et al.* [123], and it is available at [13].

7.1.3 LTE-Sim

The tool is a stand-alone software that simulates user mobility scenarios, with many users and many cells. It comes with an accurate representation of both downlink and uplink communications, and it also replicates the handover procedures when users transit from a cell to another one. It is also equipped with several traffic generators, which characterize the way in which the users relate with the LTE network.

Unfortunately, the tool comes with some drawbacks as well. The main disadvantage is that the UE, eNB, MME, and S-GW are the only entities which are represented. This means that we should have implemented the remaining entities and the signaling protocols by our own. Moreover, MME and S-GW are merged together in the same entity, which implies that the signaling interface between them is not implemented.

Secondarily, the tool has no implementation of the NAS protocol, which is extensively used during some of the considered network events. This implies a non-negligible implementation effort to modify the implemented procedures through the injection of the missing signals.

These motivations have lead to the exclusion of this tool. The tool is presented in the work of Piro *et al.* [88], and it is available at [5].

7.1.4 LTE-EPC Network simulAtor (LENA)

LENA is an extension of the well-known Network Simulator 3 (NS-3), and nowadays it is considered an official NS-3 module. Having NS-3 a very large user community, LENA benefits of the same advantage, and being a NS-3 module, a NS-3 expert can easily use LENA as well. Moreover, it comes with detailed and extensive documentation, both for usage and further development.

It represents most of the LTE entities, like UE, eNB, MME, S-GW, and P-GW. It also represents the interfaces linking them, as X2, S1, and S11. Among the network events, it is able to simulate initial attach and (X2) handover.

The drawbacks of the tool are essentially the lack of the remaining entities and interfaces. In fact, there is no representation for PCRF, HSS, and SPR, nor of their interfaces of course. Moreover, S-GW and P-GW are merged together, and therefore the S5 interface is not implemented. There is no implementation for the Active_to_Idle and Idle_to_Active transitions as well.

A secondary negative aspect is the excessive focus of the tool on the physical and data link layers details. In fact, the system level simulation has been added just in a second moment, keeping the main focus on the lower layers in the protocols stack. Furthermore, the implementation of the handling of the network events does not match our study. We argue that the implementation in the simulator is a kind of simplistic representation, as the target of the module is not the faithful representation of these procedures.

Despite the disadvantages, at first we chose LENA for our work, as it was already including most of the features we were looking for. Nevertheless, the modification of the module according to our understanding resulted too error-prone and complicated. Therefore, after an initial utilization, we decided to reject this tool as well.

The tool is presented in the works of Baldo *et al.* [31, 32], and it is available at [4].

7.1.5 PhantomNet

Rather than a single simulator, PhantomNet is a common platform that allows LTE researchers to share the same resources, running multiple experiments at the same time. In fact, PhantomNet provides programmable eNBs and computing nodes that can be used as the components of an experimental LTE network. In their experiments, researchers are free to associate the logical entities of the LTE architecture to the physical resources that PhantomNet provides. Moreover, the platform allows them to send their experimental configurations to PhantomNet remotely, which will take care about multiplexing the different experiments on the same available hardware.

In order to use commodity hardware as components of an experimental LTE network, we need to simulate the components' functions. To this aim, PhantomNet provides three simulation interfaces: OpenEPC [9], OpenLTE [10], and OpenAirInterface [8].

OpenEPC represents the most attractive proposal, as almost all the entities we are interested in are actually represented. Moreover, the idea behind OpenEPC is exactly to move the management of the entities to a remote server, which resembles our work, and therefore it seemed suitable for our aims. Nevertheless, OpenEPC requires a licence in order to be used, and the licence is not free-of-charge. For this motivation, OpenEPC was not a suitable proposal.

OpenLTE and OpenAirInterface can be freely used without any licensing cost, but at the same time they do not provide the same flexibility of OpenEPC, since the represented entities are less than the ones of OpenEPC. Given this context, none of the available interfaces was suitable for our aims.

The tool is available at [12].

7.1.6 Open5GCore/OpenSDNCore

These tools provide the implementations of the different LTE architecture elements, which can be installed on commodity hardware or programmable eNBs. They have been created in a collaboration between Fraunhofer-Gesellschaft and the Technical University of Berlin, and their aim are similar to the ones supporting PhantomNet:

inject programmability, scalability, and easiness in maintenance in the next generation LTE networks through separation between control and data planes.

The main disadvantage of these tools is that there is no precise description of what the user can do with them. In fact, their web sites provide only a very high level overview of their functionalities, without going though implementation details and hardware requirements. In order to access to the actual implementation, or simply to more detailed documentation, a licence is needed. Therefore, we have decided to exclude them from our list of available simulators.

The tools are available at [7] and [11].

7.1.7 LTE Simulator (Sandu *et al.*)

The tool is a simulator that reproduces faithfully the procedures triggered by three network events: initial attach, X2 handover, and detach. It is based on the OMNet++ framework, and it has a twofold aim: provide a graphical representation of the signals flow, and trace all the signals in a .pcap file, which may be analyzed through Wireshark. Moreover, it represents all the network elements of the LTE architecture we are interested in, except for the SPR.

Nevertheless, the tool does not provide any implementation of the procedures triggered by the UE state transitions, nor it is able to reproduce the S1 handover signals flow. Furthermore, there is no mean to measure time requested, latency, or any other performance value, as the tool has been conceived mainly for educational purposes, and therefore the primary aim is to show how the signals flow in the architecture.

Even if the tool has not been used for the analysis part of the work, it has been extremely useful in understanding the initial attach and X2 handover procedures. The tool is presented in the work of Sandu *et al.* [118].

7.1.8 nwEPC

The tool represents some entities of the core network, modelling MME, S-GW, and P-GW, supporting both control and data plane channels among them. One of the key ideas of the tool is the flexibility, since the researcher can decide how many workstations he wants to use in the experiments: he can compress everything in one single machine, he can split MME and S-GW/P-GW in two machines, or he can even add other gateways as preferred.

The final aim of the tool is to show how packets generated from a host in the Internet reach the UEs and vice versa. Please notice that the tool implements the UEs together with the MME, skipping completely any implementation of the RAN. The tool does not provide any implementation for HSS, SPR, and PCRF. Consequently, the interfaces linking them with the remaining LTE entities are not implemented as well.

Given this scenario, the tool is not suitable for our aims. In fact, in the handling of the network events we are interested in, most of the signals target (or depart from) the eNB, and therefore its implementation and the implementation of its interfaces are absolutely needed. The tool is available at [6], and it is reported in the work of Lindholm *et al.* [74] as the tool with which the ideas of the paper were being implemented.

7.2 Looking for LTE Traces

Once realized that none of the examined simulators was suitable for our needs, we decided to analyze the proposed model using the second approach. Regardless the approach selected, we needed some values as input for the simulator or for the analytical model, which previously we have called input frequencies. Now we want to define precisely which were our requirements on these input data.

The study on the 4G architecture and the refactoring process allowed us to gain the factors we have reported in the previous analysis sections. These factors express the frequency with which each interface is used in each network event. Summing them up properly allows to gain the total number of signals needed for handling each network event, as well as the total number of signals that go through a single interface in the handling of all the network events considered. The most important results are the factors related to the single network events, as they express the signals needed for handling each one of them, and therefore they directly represent the signaling storm problem. Being event-specific factors, we represent them using the signals/event unit of measure. In few words, for both the current LTE architecture and the proposed model, we are able to declare how many signals are needed in handling a single event among the ones considered.

In order to improve the analysis, we wanted to compare the current LTE architecture with the proposed model in a real scenario, which is, we wanted to understand how the proposed model would work under a typical usage pattern. A typical usage pattern is firstly defined through a set of parameters, which are the geographical area we are considering, the LTE-enabled population size, the timespan we are taking into account, the number of base stations, and so on. Once the context is defined, the usage pattern contains the frequencies with which each one of the network events considered occurs. Having these frequencies allows to multiply them with the single-event factors, gaining the total number of signals that the cellular network has actually required for the handling of these events. This allows to compare the architectures from the signaling storm perspective.

Therefore, we were looking for real LTE traces that express the frequencies with which the considered network events have occurred, given a well-defined context (*i.e.* which is the population size, the geographical area considered, and so on). The following sections describe the difficulties we have experienced in obtaining them, and the trade-off we have adopted eventually.

7.2.1 The Research Process

We have started looking for papers in which the authors report the usage of real LTE traces. Then we contacted the corresponding authors in order to ask for an access to their resources. Unfortunately, all of them have declared that actually ISPs or huge telecom companies are the owners of the LTE traces, and therefore they are not freely accessible.

We have also looked for LTE datasets directly in the Internet. The only resource we found out is Crawdad [1]: it is a freely-accessible archive of wireless traces collected by researchers or radio amateurs. The variety of traces is really huge, as the dataset accepts traces from all the different wireless technologies. Unfortunately, its section related to cellular networks has few traces, and most of them do not involve LTE, but they rather consider older technologies. The only traces about LTE networks do not come with the desired data associated: in fact, they are more focused on lower

level details, like transmission power, rather than on the system level aspects we are interested in.

This tremendous lack of LTE traces for the research community comes essentially because of two factors. The first one is that they are difficult to gain. In fact, we need a whole cellular network in order to record data related to attaches, detaches, handovers and states transitions. As researchers usually cannot afford the deployment of a cellular network for their purposes, they have to open collaborations with ISPs or telecommunications companies in order to study system-level aspects, and these huge organizations prefer to keep the collected data private. In this way, they are able to gain the benefit from the research work, without providing additional information to competing companies.

The second motivation is that the sharing of LTE traces by big organizations may result in privacy problems for them in the future. Even if the traces are anonymized, a company is not sure if in the future some new techniques will allow to infer private information from sources that were supposed to be safe. In other words, a vulnerability in the anonymization technique could be discovered, leading to information leakage on the freely-distributed LTE traces. Therefore, a company prefers to keep the traces private, in order to avoid completely this problem.

Given this complicated context, we decided to adopt a trade-off solution. Many of the documents reporting the usage of LTE traces actually describe the content of the LTE trace in an extensive way, especially through charts. Therefore, we decided to use the data reported by these documents in order to perform our analysis. In few words, instead of accessing directly to the LTE datatsets, we use their description in the documents that report their usage. In the following section, we report which are the documents we have selected for our analysis, and how the data in them have been adapted for our purposes.

7.2.2 The Data Sources

We have selected four documents suitable for our needs: Jin *et al.* [70], Widjaja *et al.* [124], the book of Metsälä [76], and a white paper from Nokia [80]. Since most of the sources do not present the data in a form suitable for our needs, we have decided to further select the sources whose data required less transformations. In fact, the transformation process is likely to inject errors in the data, and therefore we prefer using the data as it is presented in the document if possible. Two documents have passed the selection: Jin *et al.* [70], and the book of Metsälä [76].

7.2.2.1 Adapting the data

The two selected sources present their own data with a certain unit of measure. For example, Jin *et al.* [70] present the data using events/second as unit of measure; in the white paper of Nokia [80], the authors rather use events/day/subscriber as unit of measure. Instead of trying to uniform the unit of measure across the different works, we decided to keep the unit of measure of each single work and to perform the comparison using it. This is mainly because of two reasons.

The first one is that converting the values in different units of measure is just a matter of multiplicative factors. For example, if we want to express the data in [70] with the same unit of measure used in the white paper of Nokia [80], then we have to pick the values from the first work, to multiply them by 3,600 and to divide them for the population value that we are considering. What we are gaining is only uniformity

in the unit of measure across different works, but no added value to the analysis. Therefore, there is no particular attraction in performing these conversions.

The second one is because our work is providing values expressed as signals/event. In fact, considering the network events selected, we have first identified how many signals are needed in the current LTE architecture, and then we have provided the same values for the proposed model. Therefore, in order to perform the comparison, we just need some values related to the the frequency of each single event. It does not matter if we are considering the total occurrences of the events, the occurrences every second, or the occurrences every second for each subscriber. We simply need values which are event-specific that we can use together with our computed signals rates.

Therefore we transformed the data from each source in order to get a value for each one of the network events considered. The value expresses how much frequently the associated event happens, regardless if the frequency is relative to some other quantity (time or population, for example). Once obtained the values, we performed the comparison against three different implementations of the proposed model, which are:

- every entity mapped to a different physical device (Impl 1);
- RAN Control and Core Control coalesced in the same physical device (Impl 2);
- RAN Control, Core Control and eNB Data coalesced in the same physical device (Impl 3).

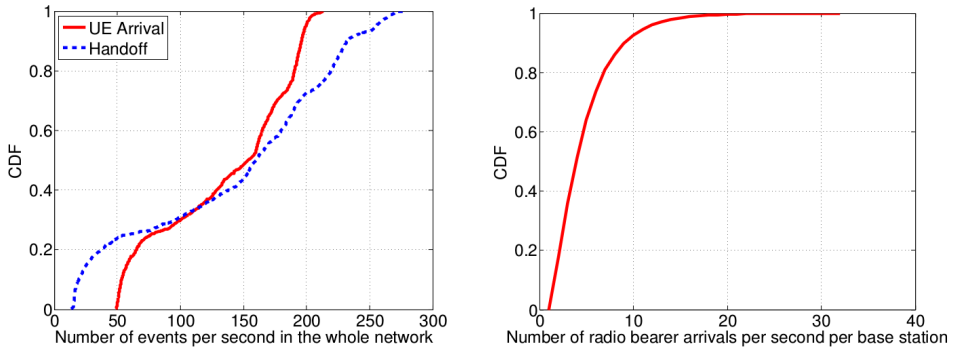
The first implementation represents the worst-case scenario of our proposed model, showing which are the drawbacks of adopting this approach. The second implementation matches the idea of having a centralized, single controller, while the third one shows the benefits of shifting the control plane to the RAN.

7.2.2.2 SoftCell: Scalable and Flexible Cellular Core Network Architecture

The authors of the paper report the usage of a huge LTE trace which has been collected by an ISP during one week in January 2013. The geographical area considered is the coverage area of approximatively 1,500 base stations, and the number of devices (UEs) considered is about 1 million. The data of the LTE trace is provided to the reader though charts: in particular, the charts in Figure 7.1a and 7.1b are the most relevant for our interests.

The first one shows the CDF of the number of initial attach events (in the whole network per second), and the CDF of the number of handover events (in the whole network per second). We proceeded as follows: we picked the 99th percentile value and we considered it as the reference value for that particular event. In order to get some values specific for each kind of handover (X2 and S1), we used the ratios expressed in Table 7.1 to split the single handover value in two meaningful values.

The second chart shows the CDF of the number of radio bearer arrivals per second per base station. We interpreted a radio bearer arrival as the activation of a default radio bearer: in this way, the radio bearer arrivals can be mapped to both initial attach events and idle_to_active transition events. Similarly to the previous case, we picked the 99th percentile value and we used the rates expressed in Table 7.1 in order to get meaningful values for each one of the remaining events considered. Please notice that, in order to remove the initial attach fraction of events in the value picked from the



(a) Events in the whole network. (b) Radio bearer arrivals per base station.

Figure 7.1: **Charts picked from [70].** *From the first one, we derived the input frequencies for initial attach and handover events, while from the second one we extracted the input frequencies for the state transition events.*

Ratio	**Value**
$\frac{Active\_to\_Idle\ transition}{Initial\ attach}$	68
$\frac{Idle\_to\_Active\ transition\ (UE\ triggered)}{Initial\ attach}$	38
$\frac{Active\_to\_Idle\ transition}{Idle\_to\_Active\ transition}$	1
$\frac{Idle\_to\_Active\ transition\ (UE\ triggered)}{Idle\_to\_Active\ transition\ (Network\ triggered)}$	1.27
$\frac{X2\ handover}{S1\ handover}$	40

Table 7.1: **Ratios in the frequencies of different network events.** *We used the ratios to split cumulative values in event-specific ones. The values for the ratios are picked from the book of Metsälä et al. [76].*

second chart, we were forced to multiply the value with the number of base stations (i.e. 1,500) to get the number of radio bearer arrivals per second in the whole network.

Summarizing, the input values for the comparison are expressed as events per second in the whole network. Table 7.2 reports the final input values.

Event	**Frequency**
Initial attach	214
Active_to_Idle transition	50,786
Idle_to_Active transition (UE triggered)	28,413.31
Idle_to_Active transition (network triggered)	22,372.69
X2 handover	273
S1 handover	7

Table 7.2: **Input values from [70].** *The unit of measure of the frequencies is events per second.*

Please notice that we have implicitly made two assumptions. The first one is that the 99th percentiles are considered as normal rates. This is a pessimistic assumption, as the 99th percentile value expresses the value under which all the occurrences fall, and therefore it represents the worst-case scenario.

The second one is that initial attach and Idle_to_Active transition events can only trigger radio bearer activations. We made this assumption as we were focusing only on default bearers, which are created and destroyed during some of the network events considered, as initial attach and the two opposite transitions.

7.2.2.3 LTE Backhaul - Planning and Optimization

This book reports techniques for improving the LTE core network in order to meet the trends of customers. In particular, the authors provide data related to a control plane traffic model which they use to represent the current demand in LTE networks. The model used was particularly suited for our needs as it has the desired level of granularity. In fact, it comes up with a frequency value specific for each one of the network events considered. This means that we did not need to perform any conversion, but we could rather use the values as they are provided.

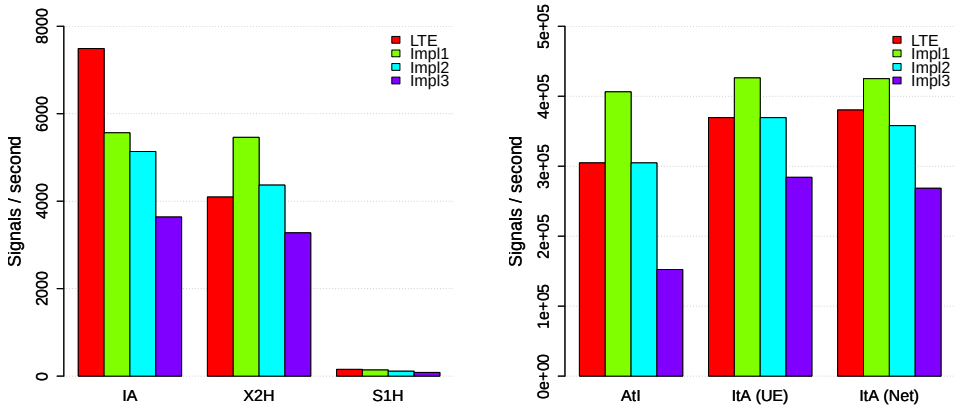
The values are expressed as events per busy hour per subscriber per base station. This implies that we can instantiate them as preferred, in the sense that we can decide the context information. As the authors of the book suggest, we can operate with these values using the number of base stations, the population size, and the time span we prefer. This flexibility has the drawback of bringing a loss of concreteness, as it is hard to imagine the provided values as suitable for all the possible scenarios. Therefore, we decided to perform the analysis directly using the values provided rather that trying to use the context information from the previous paper to uniform the values.

Table 7.3 reports the input values used for the analysis.

Please note that the book provides a value for 'Service request' events (34) and a value for 'Paging' events (15). 'Service request' corresponds to the UE-triggered Idle_to_Active transition, and 'Paging' corresponds to the signals exchanged in the LTE network in order to notify the UE of incoming downlink packets. Since a network-triggered transition is the concatenation of a paging procedure and a UE-triggered transition, we interpreted the data as follows: from the whole number of service request, we have removed the ones that have been triggered by the network in order to

Event	Frequency
Initial attach	0.5
Active_to_Idle transition	34
Idle_to_Active transition (UE triggered)	19
Idle_to_Active transition (network triggered)	15
X2 handover	8
S1 handover	0.2

Table 7.3: **Input values picked from [76].** *The unit of measure of the frequencies is events per busy hour per subscriber per base station/site.*



(a) Initial attach and handover events.

(b) Transition events.

Figure 7.2: **Comparison between the current LTE architecture and the different implementations of the proposed model (data from [70]).** *While the first implementation (Impl 1) adds a significant overhead on the most frequent events, the second implementation (Impl 2) perform as good as the current LTE architecture. The third implementation (Impl 3) definitively outperforms the current state-of-the-art.*

get the ones that have been triggered by the UE autonomously. Therefore, we have 15 network-triggered transitions, and $34 - 15 = 19$ UE-triggered transitions. Please note also that the lack of the context information makes impossible to define precisely the extendibility of our conclusions.

7.3 Comparison with the Current LTE Architecture

In the following, we present the results of our analysis. We used the input values of the two sources together with the factors obtained from the previous work to compare the current LTE architecture with the different implementations of the proposed model.

7.3.1 First Data Source

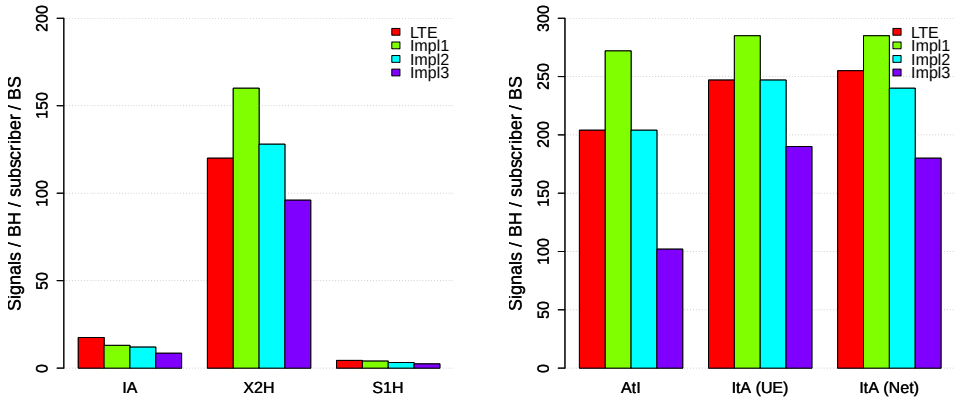
The charts in Figure 7.2 show the comparison between the current LTE architecture and the selected implementations of the software-defined model. The performance of the naive implementation are bad, as the benefits that it brings are limited to the initial attach and the S1 handover. Moreover, S1 handovers are so infrequent in this scenario that the benefits associated are almost negligible. The implementation implies an additional overhead as well: all the most frequent events are considerably heavier from the signaling perspective. The overhead on the Active_to_Idle transition events represents the worst-case, since the naive implementation requires around 100.000 signals per second more than the original LTE architecture. We can conclude saying that this implementation may result beneficial only in a context in which the dominant network event is the initial attach.

The second implementation performs similarly to the current LTE architecture. Its performance are better considering initial attach, S1 handover and Idle_to_Active transition (network triggered) events, but it adds some overhead on the X2 handover procedure. This is mainly because this implementation does not use the X2 interface for control plane messages, but all the signals have to pass through the external control entity, whereas the current LTE architecture allows the base stations to partially auto-configure the reaction to the event (if a link between them exists). We argue that putting intelligence in base stations should not be a default choice, but rather a possibility: as the third implementation shows, we can actually put the control plane in a base station, and we can choose in which one according to our availabilities and needs. This approach allows to lower the costs of the base stations, since they are reduced to remotely-controlled radio devices. In conclusion, the second implementation does not bring any particular benefit from the performance perspective, but we are gaining something on the architectural side: the devices belonging to forwarding and storage levels are remotely configurable and have no cleverness, and all the intelligence is in an affordable, remote, dedicated server.

The charts in Figure 7.2 show how shifting the control plane to the RAN outperforms the current LTE architecture. In fact, the third implementation brings benefits on the handling procedure for each one of the network events considered. This is particularly evident considering initial attach and Active_to_Idle transition, in which the reduction of the signaling demand is about 50%. The significant contraction in the signaling load is because, in the current LTE architecture, the eNB often acts as a by-pass between the UE and the control plane of the cellular network. We can notice it during the security procedures (*i.e.* authentication, NAS security setup), in which the eNB acts simply as a relay. This implementation cuts half of the signals needed in these situations, as the cost of a signal from the control level to the eNB (and vice versa) drops to zero. Since the signaling load is reduced in each one of the considered events, the implementation results suitable for every scenario in which a cellular network is needed.

7.3.2 Second Data Source

The naive implementation of the proposed model brings a significant increase in the signaling demand of Active_to_Idle transition events and X2 handovers (Figure 7.3a and Figure 7.3b). The additional load is particularly consistent, since the implementation asks for up to 68 signals more for each subscriber during busy hours just for handling one of the network events. Similarly to the previous case, the only benefits



(a) Initial attach and handover events.

(b) Transition events.

Figure 7.3: **Comparison between the current LTE architecture and the different implementations of the proposed model (data from [76]).** *Similarly to the previous analysis scenario, the first implementation (Impl 1) worsens the signaling issue, while the second and the third implementations (Impl 2 and Impl 3) bring architectural advantages and reduce the signaling load.*

come on initial attach and S1 handover procedures.

The second implementation requires a signaling load similar to the one requested by the current LTE architecture. There are some small reductions of the load considering initial attach, S1 handover, and especially Idle_to_Active transition (network triggered). The X2 handover is still the procedure which suffers more for the adoption of the proposed model. We can explain this phenomenon in the same way as we have done previously: the second implementation does not take advantage of the X2 interfaces for control signaling, but it rather use them just for data exchange. Instead, the current eNBs mix control and data planes, and this allows to partially configure the handover procedure among themselves. The main aim of software-defined networking is not the improvement of the performance, but rather the simplification in management of the network. When the management is easy, it is easy to boost up the performance as well.

As expected, the signaling load of the third implementation is smaller than the one requested by the current LTE architecture. In particular, one of the most frequent network events, the Active_to_Idle transition, asks for half of the signals requested by the current state-of-the-art. In this case, a such consistent reduction of the signaling load is particularly precious as we are considering data related to busy hours, and therefore the worst-case scenarios for a cellular network.

7.3.3 Final Considerations

As the number of signals for each event is fixed, we can compute the percentage of reduction (or addition) of signals that each implementation implicates. Figure 7.4 shows the percentage difference between the current LTE architecture and the first implementation of the proposed model, in which each entity is mapped to a differ-

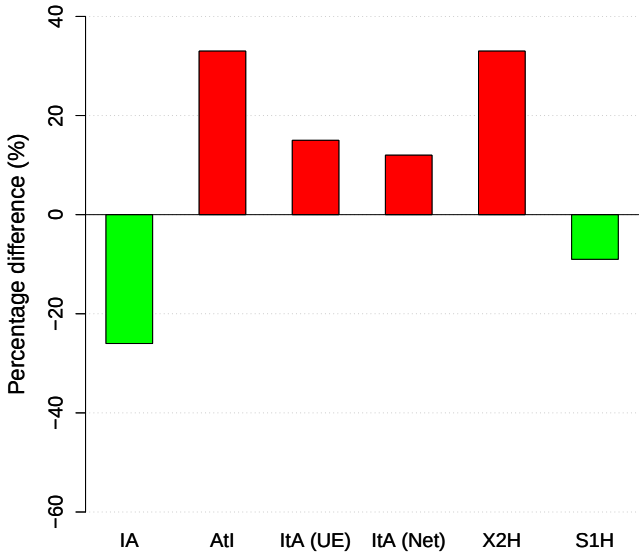


Figure 7.4: **Percentage difference in signaling load between the current LTE architecture and the first implementation of the proposed model.** *The small reductions brought on some events (initial attach and S1 handover) are not enough to counterbalance the heavy increases on the other events, which go up to +30%.*

ent physical device. We can notice how the results of the chart reflect the analysis performed with the two data sources: the implementation brings a significant reduction in the signaling load associated with initial attach events, but at the same time it implies a consistent increase in the signaling demand for events which are usually more frequent in a cellular network, like Active_to_Idle transitions and X2 handovers. As we pointed out previously, this kind of implementation could result beneficial in a context in which the initial attach is the most frequent network event. This could be the case of a sensor network in which the sensors attach to a common base station and perform their data collection till running out of energy (*e.g.* sensors in a forest).

The chart in Figure 7.5 depicts the same analysis performed with the second implementation of the proposed model, whose main focus is to reduce the costs associated to the communication between the control entities putting them together in a single physical device. The implementation improves the benefits brought by the first one, as both initial attach and S1 handover record a reduction around 30% in the signaling load. We can notice a small reduction (6%) on the number of signals associated with the Idle_to_Active transition (network triggered). Looking at Active_to_Idle transitions and Idle_to_Active transitions (UE triggered), the implementation does not bring benefits nor drawbacks, but it records still an additional overhead (7%) on the X2 handover events. Since the drawbacks of the proposal are limited to X2 handovers, the implementation is suitable for all the scenarios that involve base stations which are not directly linked with each other, or in which the mobility of the users is not the primary aspect.

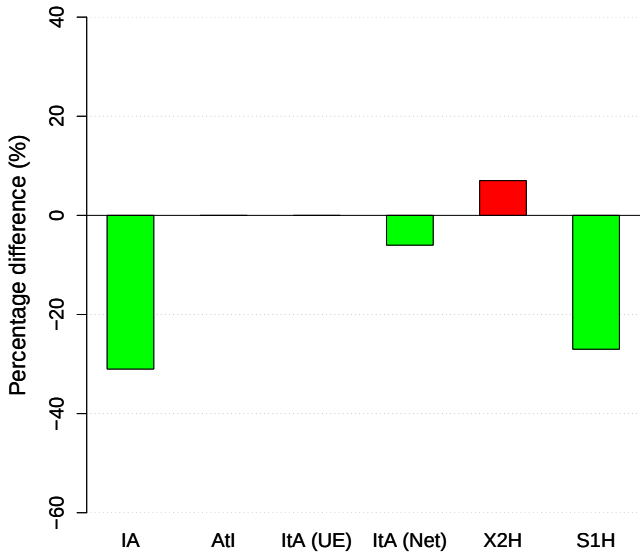


Figure 7.5: **Percentage difference in signaling load between the current LTE architecture and the second implementation of the proposed model.** *The implementation keeps the trend started by the first one further reducing the signaling load of initial attach and S1 handover events, while cutting the additional overhead associated to the other network events.*

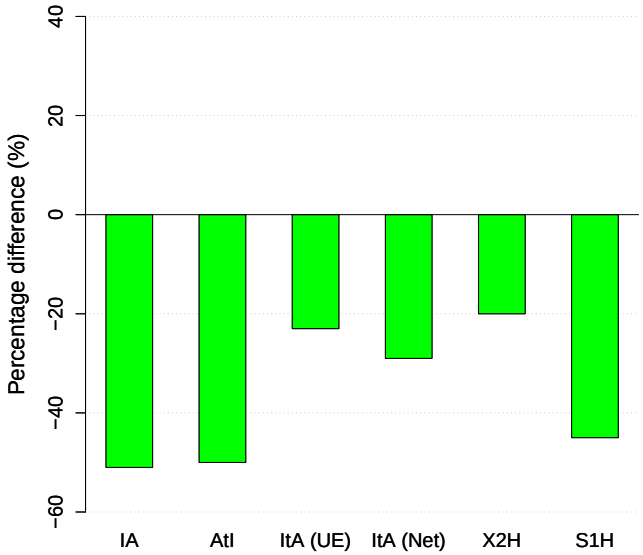


Figure 7.6: **Percentage difference in signaling load between the current LTE architecture and the third implementation of the proposed model.** *The implementation is fully beneficial, granting huge reductions in the number of signals requested, with contractions around 50% of the signaling load for some network events.*

The third implementation of the proposed model puts the whole control level close to base stations, in order to reduce the communication costs between the eNB Data and the control entities. This architectural choice improves performance, scalability and reliability of the cellular network as the chart in Figure 7.6 shows. The load associated with each one of the network events results significantly reduced, going from a minimum of 23% (Idle_to_Active transition, UE triggered) to a maximum of 51% (initial attach) percentage reduction in the amount of signals requested. Please notice that these performance values are valid just considering the base station in which the control entities are actually embedded. This means that, for all the base stations controlled by the same control entities, the performance values are the one depicted in Figure 7.5, as the control layer is in another physical device from their perspective. Nevertheless, this does not hold for handover events, as the analysis on the third implementation takes into account the signals between the remotely-controlled eNB and the locally-controlled one. Please notice also that the analysis on the handover events considers the worst-case, in which the control entities are embedded in the destination eNB (TeNB). If the control entities were embedded in the source eNB (SeNB) instead, we would experience a signal less on the S1-MME interface for each handover event. We can conclude saying that the third implementation of the model is actually able to contrast the signaling storm that is already affecting the current LTE architecture. Moreover, the application of software-defined networking grants an easier management and an affordable deployment of cellular networks.

Chapter 8

Related Work

Our work has been heavily influenced by many studies that have been performed on the same topic. Among all, the paper of Lindholm *et al.* [74] consists in the starting point of our work. The authors provide a high-level view of the state variables that are created and managed during the procedures triggered by some network events. In particular, the procedures on which we have focused in our work are the same analyzed in this paper. Moreover, the authors group the state variables in categories: the aim behind is to group variables that participate to the same aim, and that are also managed in a uniform way. This means that variables belonging to the same category are approximatively created by a same entity and sent to a same group of entities. This has allowed the authors to highlight the producer-consumer relationships between the entities of LTE networks, paving the way to their transition to the Information Centric Networking (ICN) paradigm. The final aim of the work is to leverage the identified relationships in order to refactor the LTE architecture moving the producers of state variables close to their consumers. Similarly to our work, the authors leverage on NFV and SDN to propose their refactoring: the first is used to abstract and modularize the functionalities provided by the network, while the second allows an efficient management of the identified modules. They also propose the coalescing of the control entities in the same logically centralized controller, but they keep a separation between RAN and core network in their management entities, in a similar fashion to our final model.

Generally, the works that propose refactoring of the current LTE architecture can be subdivided in two categories: the ones that focus on the mobile core, and the ones that focus on the RAN. The work of Jin *et al.* [70] belongs to the first category: it proposes a core network composed by simple switches in order to solve the issues affecting the P-GW of currently deployed LTE networks. In fact, the P-GW is a monolithic and inflexible device providing several functions: this results in three main problems. First, the whole traffic of the cellular network has to pass through it, but while this traffic can increase and decrease, the P-GW cannot scale with it. Second, if a mobile operator needs just one functionalities of the many provided by a commercial P-GW, it is forced to buy the whole gateway, as the P-GW lacks of modularity. Third, there are no possibilities for mobile operators to mix the functionalities of different vendors, as the behaviour of the device cannot be freely customized. Therefore, the authors propose to substitute this single device with software-defined switches, whose control plane is composed by an arbitrary number of middleboxes. Each middlebox takes care of one function (*e.g.* firewall, DPI, and so on), so they instruct the switches

accordingly. As the rules that each switch has to take into account may increase exponentially, the authors propose a multi-dimensional aggregation technique, such that the number of entries that each switch has to handle stays small. Moreover, the authors propose to shift most of the computational complexity (*e.g.* packets classification) to the RAN, in order to keep the core dumb: following the same idea, please notice that in our work the P-GW still enforces the policy rules, but their generation is in charge of the RAN.

Instead, the work of Gudipati *et al.* [65] focuses on the refactoring of the Radio Access Network. The authors start presenting the many problems that arise from a set of loosely coordinated base stations. In fact, base stations mostly take autonomous decisions, or adopting techniques as Self-Organizing Network (SON) that include just few base stations. In this context, it is nearly impossible to perform decision on a global view of the base stations: these decisions include load balancing , interference management, and throughput maximization. Therefore, the authors propose to abstract all the base stations of a same geographical region as a single base station, composed by a single controller and several radio elements. Please notice that this idea maps 1-to-1 with our idea of the single RAN Control for all the dumb base stations. Moreover, in order to find a trade-off between dumbness of the radio elements and latency introduced by the splitting of control and data plane, the authors propose to perform in each single base station the decisions that do not affect the neighbour base stations. Therefore, this idea includes the execution of the lower level protocols directly in the radio elements, which is suggested by our work as well.

Some studies are focused on specific elements of the LTE architecture instead. For example, the work of Banerjee *et al.* [33] proposes to refactor the Mobile Management Entity (MME) in order to solve the signaling storm phenomenon. In fact, as reported in [29], the MME is the component which suffer most these storms, as most of the control plane decisions during the procedures triggered by the network events are in charge of it. Moreover, the MME has not been designed with elasticity in mind, and this is a big issue because it cannot evolve together with the subscribers' needs. Therefore, the authors propose a new MME architecture, by splitting it in two parts: a front-end, which consists in the classical MME interface, and a back-end, which dynamically scales leveraging load distribution techniques as consistent hashing [71]. Please notice that the refactoring approach resembles the one we adopted. In fact, we have kept the same interface with the LTE-enabled devices, while we have changed the architecture behind the scenes. Similarly, this work provides the same MME external interface, while changing the internal behaviour: the objective is to keep the rest of the network elements as-is. This choice allows an incremental deployment of the proposed MME architecture, and this consists in a key factor in the actual implementation of the proposed idea. In particular, the new MME architecture is composed by a dynamic number of MME Load Balancers (MLBs) and MME Processing entities (MMPs). Each MLB chooses the MMP to attach, which corresponds to the real brain of the network. Instead, the many MMPs use consistent hashing among them for load balancing and replication of the subscribers' information.

On the same lines, the work of An *et al.* [30] proposes a new MME architecture, called dMME. The motivations behind are the same of the previous work: since the MME has to be a reliable component able to handle all the signals coming from the customers, there is usually an over-provisioning of this resource, which implies a high CAPital EXpenditure (CAPEX). Therefore, the refactoring of the MME has a twofold aim: allow to scale with the load, without requiring specialized and expensive hardware. The authors propose a distributed MME architecture, composed by elements

functionally equal. The elements may consist in servers of a cluster, as well as eNBs hardware: the idea is to rely on commodity hardware, or already-available hardware which is partially unused, in order to make the costs dropping. The distributed architecture has the advantage of improving the fault tolerance: in fact, since all the components are equal, it is easy to substitute a component with another one. This allows also to limit the geographical scope of the fault, since just a single of the many component may be damaged: this is not possible in the traditional MME architecture, since we have a single physical box for it, and therefore a fault compromises the whole network. The authors also report lightweight mechanisms to transfer the user status from one component to the others, in order to enable easy load balancing and smooth handovers. Similarly to our work, the paper suggests to leverage the user locality (*e.g.* most of the users tend to move just in a small neighbourhood) by shifting the control plane as close as possible to the users. This brings the obvious advantage of reducing the latencies in the processing of signaling messages.

Instead, Basta *et al.* [34] focused on the application of NFV and SDN on the mobile core gateways, namely S-GW and P-GW. In particular, the authors focus on estimating number and geographical position in U.S.A. of refactored gateways in order to provide the current LTE coverage, but keeping the delays introduced by SDN and NFV limited at the same time. While we envision SDN and NFV applied together, their perspective about the implementation of the two networking paradigms is different. From the authors' point of view, applying NFV on a gateway means shifting both control and data planes on a datacenter, keeping the network element simply to relay packets between the network and the datacenter. Instead, applying SDN consists in shifting to a datacenter the control plane only, keeping the user plane in the network element. In this context, the complete virtualization of a gateway implies a significant increase in its packet processing time, as all the packets are processed in software, while the increase with software-defined decomposition is smaller, since the data plane is still handled on hardware. We argue that a NFV-enabled gateway has to provide a SDN interface as well, such that the forwarding of the packets to the datacenter is limited as much as possible. Indeed, the SDN interface is enough to configure most of the basic services on the packet flows (*e.g.* redirection, forwarding, dropping) : in this case, the packets do not need to be processed in the datacenter, so no indirections are needed. Therefore we redirect the packets to the datacenter only when special services have to be applied on them, as Deep Packet Inspection (DPI) or optimizations. This allows to limit the worsening of performance linked with the introduction of SDN and NFV in LTE networks.

The work of Moradi *et al.* [78] proposes a higher-level refactoring in order to solve the problems affecting each different LTE network. In fact, the authors describe a cellular Wide Area Network (WAN) architecture composed by controllers that can be configured hierarchically. They start depicting the problems of the current LTE networks: the increasing mobile traffic results in scalability issues for the data plane, while the signaling storms are the manifestation of the scalability issues of the control plane. Moreover, they argue that the current organization of the LTE networks in large and rigid regions contributes to the issues as well. The authors describe the separation between the control and data planes as the way to face these scalability issues. The control plane has to be moved to the cloud, or in any location in which the computational power is plentiful. Instead, the data plane has to be composed by dumb devices, which maintain as less state as possible. In addition, a nation-wide view of the entire set of LTE networks opens the opportunity for global optimizations. In order to achieve this, the authors propose a tree structure composed by distributed controllers.

Each controller is a modular component, that can manage a single network element, a whole LTE network, or a set of LTE networks: this is possible through abstraction and recursion. In fact, each controller directly manages a set of devices, and abstracts another set of devices. The devices abstracted by a controller are considered as normal devices by other controllers, which can manage them directly. Therefore, a controller can abstract a whole network of directly-managed switches as a single switch. Another controller can manage directly the virtual single switch: the controller providing the abstraction takes case about the translation of the high-level rules into low-level ones. This enable a wide-area management of the cellular networks using relatively easy components used in a recursive fashion.

Finally, the work of Kempf *et al.* [72] presents an extension of the OpenFlow protocol in order to enable the handling of GTP packets. The matching between the rules in OpenFlow tables and the header of packets is increased with two fields: one of 2 bytes, consisting in the GTP header flags, and one of 4 bytes, consisting in the GTP Tunnel Endpoint Identifier (TEID). While the TEID uniquely identifies the GTP tunnel, the header flags consist in information about the routing path that the packets of the tunnel have to follow. The ability of eNB, S-GW, and P-GW to handle GTP packets through OpenFlow brings several advantages. First, GTP control packets (GTP-C packets) can be easily forwarded to the controller, while GTP data packets (GTP-U) are handled locally. Second, since these network elements are able to encapsulate and decapsulate GTP packets, these ones can be presented to a chain of services in the datacenter as simple IP packets: therefore, there is no need to create network services specialized for GTP packet flows. Please note that this extension applies only to eNBs considering our work. In fact, in our model the eNB represents the only network element in which control plane and data plane messages go through at the same time. This happens because the control plane messages from and to the UE have necessarily to flow through the eNB, and once reached the eNB, they are forwarded to the control layer, where they are processed. Therefore, excluding the eNB, the network elements belonging to the forwarding layer are not supposed to handle control plane messages. Nevertheless, this work represents a key contribution in the transition from currently-deployed network element to software-defined ones.

Chapter 9

Conclusion and Future Works

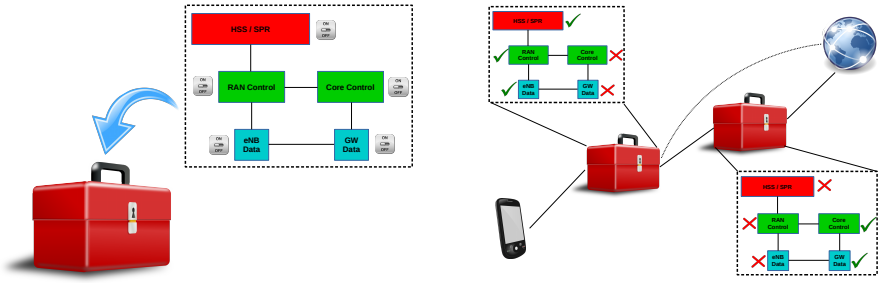
In this chapter, we summarize benefits and drawbacks of our work. First, we depict which are the outcomes of our analysis and refactoring, and then we describe some ways in which them may be leveraged. Finally, we express which are the limitations of our work, and how they can be used as motivations for future improvements of the ideas presented here.

9.1 Outcomes

The analysis performed in the previous chapter shows that different implementations of the proposed model bring different advantages from the signaling perspective. In particular, the coalescing of the control plane inside a base station does not bring only a significant reduction in the signaling load, but it also optimize the usage of the base stations, as the wasted processing capacity is now leveraged. In this context, we can say that we have reached our initial aim: we have gained an architecture that allows to reduce the signaling storm phenomenon without requiring modification on the user equipment's side.

Nevertheless, we argue that the avoidance of signaling storms is just one of the many advantages that our architecture can bring. In fact, we strongly believe that the true outcome of the proposed architecture consists in its intrinsic flexibility. This has been gained through the identification of precise software and hardware requirements: as we have seen, the software requirements may be met by an arbitrary number of computing entities, while the hardware requirements are met by a number of hardware devices that depends on our availabilities. This allows to customize the mapping between architectural entities and physical devices in order to meet the requirements coming from different verticals.

For example, consider a sensor network composed by LTE-enabled sensors. These sensors have a fixed geographical position, and they carry out a very simple task: they attach to a base station, and they collect and send data till they run out of energy. In this context, the only procedure that may influence the performance of the system is the initial attach. Therefore, we can design our architecture in order to minimize the signals requested by the initial attach procedure, without caring about the impact for the state transitions or for the handover procedures. We have seen that coalescing of control plane in a base station can result an excellent choice to reduce the signals required during initial attach.



(a) The *Box* schema. (b) A possible configuration of two *Boxes*.

Figure 9.1: **The proposed model compressed in a single Box.** *In the deployment scenario, the first Box is connected to the Internet for fault tolerance purposes.*

Similarly, consider a huge mining site with a LTE-based monitoring system. In this scenario, each mining wagon may be a LTE-enabled device that has continuously to send some data to the monitoring console. The procedures that matter in this case are the handovers, as the wagons attach only one at the beginning of the day, and they do not have any strict energy requirement, so the state transitions are not implemented. Again, we can customize our implementation in order to minimize the signals during handovers, without caring about the remaining procedures. This can be achieved using a device with both eNB and gateway capabilities, such that the buffering of the downlink packets can take place directly in it, and start/stop messages are needed no more.

These examples show that the proposed architecture can be tailored to the specific needs of the customer, and the avoidance of signaling storms is just one of these many needs. We have reported the ideas of our work in a scientific article, which have been submitted to a prestigious conference. The details of the submission can be found in Appendix B.

9.2 The Network-in-a-Box Idea

The unbridled flexibility reported in the previous section lead us to the ultimate idea of compressing all the entities of our architecture in a single box, henceforth referred as the *Box*. The idea is to encapsulate all the architectural entities in a single physical device, as depicted in Figure 9.1a. Each entity is equipped with a switch, that allows to enable or disable the associated component. The aim of the *Box* is to represent the basic building block of each cellular network: in fact, a network can be deployed through a single *Box*, as well as through several *Boxes* properly configured. This would enable the organic growth of the network envisioned by Thompson *et al.* in their work [122].

Consider the deployment scenario depicted in Figure 9.1b. The first *Box* acts as the eNB Data, the RAN Control, and the HSS of the whole cellular network, while the second *Box* executes Core Control and GW Data modules only. This is possible thanks to the switch with which each entity in each box is equipped. Please note that the powered-off components can be activated in case of faults: indeed, the first *Box* has a spare Internet connection, such that if the second *Box* suddenly fails, Core Control and GW Data can be activated in the first *Box*, keeping active the LTE network.

There are several deployment scenarios in which a network in a single physical component would be suitable for the specific needs. As reported in [81, 82], these include mining sites, public safety operations, Internet provisioning in remote locations, and enterprise private communications. Similar projects aim to provide a fully-fledged LTE-network through a single physical device as well [2, 3].

Nevertheless, the idea of putting a whole database of subscribers' information in a single, practical device could seem unfeasible. Actually, the storage layer could be replaced with a cache: in this context, the device can use an Internet connection to minimize the accesses to the true storage layer caching the information that most likely are going to be required by the LTE network. This opens opportunities to study caching techniques aimed to minimize the accesses to the actual database.

9.3 Limitations and Future Works

In this section, we aim to express clearly the limitations of our work that we have identified. The first aspect to note is that the whole work is focused on the procedures we have selected, but no considerations are made about the remaining procedures carried out by current LTE networks (*e.g.* Tracking Area Update, Detach, and so on). This has been decided for two reasons: the first is because the selected network events are among the most frequent ones in current LTE networks, and the second is to limit the scope of our research work. There are two ways of looking to this: one one hand, this results in an implementation issue, as the model cannot be materialized as-is; on the other hand, it opens a perspective for a future research work, which is the extension of the proposed model to the remaining procedures.

A huge drawback of our work is that retrocompatibility with older standards, as 2G or 3G, is not taken into account. In fact, retrocompatibility always represents one of the driving principles in every standardization process, and the aim is clear: there is the will to keep operating all the obsolete devices that use older standards, since a sudden stop in their functioning would not be appreciated by their owners. In this context, we have limited our retrocompatibility with currently deployed LTE-enabled devices, as they can benefit of the proposed architecture without requiring any modification. Nevertheless, no considerations are made about previous standards, and also the analysis of the functions carried out by each networking element has not touched tasks related with retrocompatibility. Therefore, this greatly limits the implementation perspective of the proposed model. Given this scenario, there are two considerations. First, this issue can represent a future research work: in the same way as the model can be extended to the remaining procedures, it can be extended in order to provide retrocompatibility with previous standards as well. Second, this issue does not exist if our deployment scenario consists of LTE-enabled devices only. This may be taken into account in the design of a system based on cellular networks: if we already know the retrocompatibility issue, we can study our customized solution in order to avoid it from the beginning.

Finally, our perspective about the current LTE architecture is that every logical entity is mapped to a separated physical device. We assumed this as the only reference documents available are the 3GPP specifications, but the actual implementation of the standards may be specific for each different vendor. Therefore, as there are no standard implementation schemas, we have assumed the worst case, where every entity consists in a physical device by its own. Actually, this may not correspond to the reality: many vendors implement the S-GW together with the P-GW [28], and the HSS together with

the SPR as well. Nevertheless, the lack of official and standard information has lead us to the performed worst-case analysis, which matches a conservative approach to the problem.

Appendix A

Detailed procedures

In this chapter, we describe how the state of the network change during each one of the examined procedures of the current LTE architecture. In particular, we examine the information created, carried, and destroyed by each signal in each procedure. A signal creates a state variable if that state variable is created in the destination network element after the reception of the signal. A signal carries a state variable if the state variable is available in the destination network element after the reception of the signal. A signal destroys a state variable if the state variable is destroyed in the destination network element after the reception of the signal. Please note that the network elements may relay some information for neighbour network elements. Therefore, in these cases the state variables are labelled with 'relay'. We strongly suggest to read the following sections using the material available at [94].

A.1 Initial Attach

- Step 1: Random Access Preamble (UE → eNB)

  This message is sent by the UE to signal its will to join the network of the eNB.

  Information carried:

  None

  Information created:

  – C-RNTI (eNB)

- Step 2: Random Access Response (UE ← eNB)

  This message is sent by the eNB to regulate the access to the wireless channel.

  Information carried:

  – C-RNTI

  Information created:

  None

- Step 3: RRC Connection Request (UE → eNB)

  The message is sent to create a control plane channel between UE and eNB.

  Information carried:

None

Information created:

None

- Step 4: RRC Connection Setup (UE ← eNB)

  The message is sent to regulate the parameters of the control plane channel.

  Information carried:

  None

  Information created:

  None

- Step 5: RRC Connection Setup Complete (UE → eNB)

  The message is sent to acknowledge the channel establishment.

  Information carried:

  – IMSI (relay)
  – UE Network Capability (relay)

  Information created:

  – eNB S1AP UE ID (eNB)

- Step 6: Initial UE Message (eNB → MME)

  The message is sent to create a control plane channel between eNB and MME.

  Information carried:

  – IMSI
  – UE Network Capability
  – ECGI
  – TAI
  – eNB S1AP UE ID

  Information created:

  – MME S1AP UE ID (MME)

- Step 7: Authentication Information Request (MME → HSS)

  The message is sent to request the authentication vectors needed in the authentication procedure.

  Information carried:

  – IMSI

  Information created:

  – RAND (HSS)
  – AUTN (HSS)
  – XRES (HSS)

  - K_{ASME} (HSS)

- Step 8: Authentication Information Answer (MME ← HSS)
  The message carries the authentication vectors requested by the MME.
  Information carried:
  - RAND
  - AUTN
  - XRES
  - K_{ASME}

  Information created:
  - KSI_{ASME} (MME)

- Step 9: Authentication Request (eNB ← MME)
  The message is sent to start the mutual authentication with the UE.
  Information carried:
  - RAND (relay)
  - AUTN (relay)
  - KSI_{ASME} (relay)

  Information created:
  None

- Step 10: Authentication Request (UE ← eNB)
  The message is sent to start the mutual authentication with the UE.
  Information carried:
  - RAND
  - AUTN
  - KSI_{ASME}

  Information created:
  - RES (UE)

- Step 11: Authentication Response (UE → eNB)
  The message is sent to complete the mutual authentication with the UE.
  Information carried:
  - RES (relay)

  Information created:
  None

- Step 12: Authentication Response (eNB → MME)
  The message is sent to complete the mutual authentication with the UE.
  Information carried:

  - RES

  Information created:

  - NAS Security Algorithm (MME)
  - UE Security Algorithm (MME)

- Step 13: Security Mode Command (eNB ← MME)

  The message establishes the keys and the algorithms that are used to set a secure communication between UE and MME.

  Information carried:

  - KSI_{ASME} (relay)
  - NAS Security Algorithm (relay)

  Information created:

  None

- Step 14: Security Mode Command (UE ← eNB)

  The message establishes the keys and the algorithms that are used to set a secure communication between UE and MME.

  Information carried:

  - KSI_{ASME}
  - NAS Security Algorithm

  Information created:

  None

- Step 15: Security Mode Complete (UE → eNB)

  The message acknowledges the security settings to the MME.

  Information carried:

  None

  Information created:

  None

- Step 16: Security Mode Complete (eNB → MME)

  The message acknowledges the security settings to the MME.

  Information carried:

  None

  Information created:

  None

- Step 17: Update Location Request (MME → HSS)

  The message is sent to inform the HSS about the current location of the UE.

  Information carried:

  - IMSI

  - MME ID

  Information created:

  None

- Step 18: Update Location Answer (MME ← HSS)

  The message carries the information needed for the session establishment.

  Information carried:

  - Default APN
  - P-GW ID
  - QCI_{HSS}
  - ARP_{HSS}
  - UE-AMBR UL
  - UE-AMBR DL
  - APN-AMBR UL_{HSS}
  - APN-AMBR DL_{HSS}

  Information created:

  - EPS Bearer ID (MME)
  - P-GW IP (MME)
  - APN In Use (MME)

- Step 19: Create Session Request (MME → S-GW)

  The message is sent to start the creation of the data plane channel.

  Information carried:

  - IMSI
  - EPS Bearer ID
  - P-GW IP
  - APN In Use
  - QCI_{HSS} (relay)
  - ARP_{HSS} (relay)
  - APN-AMBR UL_{HSS} (relay)
  - APN-AMBR DL_{HSS} (relay)
  - ECGI
  - TAI

  Information created:

  - S5 TEID DL (S-GW)

- Step 20: Create Session Request (S-GW → P-GW)

  The message is sent to continue the creation of the data plane channel.

  Information carried:

  - IMSI
  - EPS Bearer ID
  - APN In Use
  - QCI_{HSS} (relay)
  - ARP_{HSS} (relay)
  - APN-AMBR UL_{HSS} (relay)
  - APN-AMBR DL_{HSS} (relay)
  - ECGI
  - TAI
  - S5 TEID DL

  Information created:

  - UE IP Address

- Step 21: EPS Session Establishment Notification (P-GW → PCRF)

  The message provides the information needed to generate the appropriate rules (PCC policies) to the PCRF.

  Information carried:

  - IMSI
  - UE IP Address
  - APN In Use
  - QCI_{HSS}
  - ARP_{HSS}
  - APN-AMBR UL_{HSS}
  - APN-AMBR DL_{HSS}
  - ECGI
  - TAI

  Information created:

  None

- Step 22: Profile Request (PCRF → SPR)

  The message requests the access profile of the subscriber to the SPR.

  Information carried:

  - IMSI

  Information created:

  None

- Step 23: Profile Response (PCRF ← SPR)

  The message carries the access profile of the subscriber.

  Information carried:

  - SDF Filter
  - QCI_{SPR}
  - ARP_{SPR}
  - APN-AMBR UL_{SPR}
  - APN-AMBR DL_{SPR}

  Information created:

  - QCI_{PCC} (PCRF)
  - ARP_{PCC} (PCRF)
  - APN-AMBR UL_{PCC} (PCRF)
  - APN-AMBR DL_{PCC} (PCRF)

- Step 24: EPS Session Establishment Ack (P-GW ← PCRF)
  The messages carries the rule(s) to apply on the subscriber's packet flows.
  Information carried:

  - SDF Filter
  - QCI_{PCC}
  - ARP_{PCC}
  - APN-AMBR UL_{PCC}
  - APN-AMBR DL_{PCC}

  Information created:

  - S5 TEID UL (P-GW)
  - TFT UL (P-GW)
  - TFT DL (P-GW)

- Step 25: Create Session Response (S-GW ← P-GW)
  The message answers to the Creation Session Request.
  Information carried:

  - UE IP Address (relay)
  - EPS Bearer ID
  - S5 TEID UL
  - QCI_{PCC}
  - ARP_{PCC}
  - APN-AMBR UL_{PCC} (relay)
  - APN-AMBR DL_{PCC} (relay)
  - TFT UL (relay)

  Information created:

  - S1 TEID UL (S-GW)

- Step 26: Create Session Response (MME ← S-GW)

  The message answers to the Creation Session Request.

  Information carried:

  - UE IP Address
  - EPS Bearer ID
  - S5 TEID UL
  - S5 TEID DL
  - S1 TEID UL
  - QCI_{PCC}
  - ARP_{PCC}
  - APN-AMBR UL_{PCC}
  - APN-AMBR DL_{PCC}
  - TFT UL (relay)

  Information created:

  - GUTI (MME)
  - TAI List (MME)
  - TAU Timer Value (MME)
  - E-RAB ID (MME)
  - K_{eNB} (MME)

- Step 27: Initial Context Setup Request (eNB ← MME)

  The message sets up the final part of the data plane channel (between UE and eNB).

  Information carried:

  - UE IP Address (relay)
  - GUTI (relay)
  - EPS Bearer ID (relay)
  - APN In Use (relay)
  - TAI List (relay)
  - TAU Timer Value (relay)
  - APN-AMBR UL_{PCC} (relay)
  - TFT UL (relay)
  - E-RAB ID
  - QCI_{PCC}
  - ARP_{PCC}
  - UE-AMBR UL
  - UE-AMBR DL
  - K_{eNB}

  - UE Security Algorithm
  - S1 TEID UL
  - MME S1AP UE ID

  Information created:

  - AS Security Algorithm (eNB)

- Step 28: Security Mode Command (UE ← eNB)

  The message establishes a secure communication between UE and eNB.

  Information carried:

  - AS Security Algorithm

  Information created:

  None

- Step 29: Security Mode Complete (UE → eNB)

  This message completes the security operations between UE and eNB.

  Information carried:

  None

  Information created:

  - DRB ID (eNB)

- Step 30: RRC Connection Reconfiguration (UE ← eNB)

  This message carries the Attach Response as answer for the NAS-layer Attach Request.

  Information carried:

  - DRB ID
  - UE IP Address
  - GUTI
  - EPS Bearer ID
  - APN In Use
  - TAI List
  - TAU Timer Value
  - QCI_{PCC}
  - APN-AMBR UL_{PCC}
  - TFT UL

  Information created:

  - S1 TEID DL (eNB)

- Step 31: Initial Context Setup Response (eNB → MME)

  This message completes the data plane channel creation.

  Information carried:

  - E-RAB ID
  - S1 TEID DL

  Information created:

  None

- Step 32: Attach Complete (UE → eNB)

  The message signals the completion of the attach procedure.

  Information carried:

  - EPS Bearer ID (relay)

  Information created:

  None

- Step 33: Attach Complete (eNB → MME)

  The message signals the completion of the attach procedure.

  Information carried:

  - EPS Bearer ID

  Information created:

  None

- Step 34: Modify Bearer Request (MME → S-GW)

  This message completes the creation of the data plane channel between eNB and S-GW.

  Information carried:

  - EPS Bearer ID
  - S1 TEID DL

  Information created:

  None

- Step 35: Modify Bearer Response (MME ← S-GW)

  The message acknowledges the completion of the the data plane channel creation.

  Information carried:

  None

  Information created:

  None

A.1.1 Considerations

We apply the following simplification in the graphical representation of the procedure in [92]:

- The value 'LTE K' is not used in any of the message exchanges, therefore it is neglected.
- 'APN' is a value owned by the UE that may be sent to the MME after the NAS Security Setup and before the Update Location Request to the HSS [16]. Then, the MME uses this information and the 'Default APN' retrieved by the HSS in order to decide the final 'APN In Use'. In our example, we are assuming that this further interaction does not take place, meaning that:
  - 'APN' is not used in any message exchange, so it is neglected;
  - There is no difference between 'Default APN' and 'APN In Use', so they can be substituted by a single value 'Default APN'.
- The values 'UE-AMBR UL' and 'UE-AMBR DL' are always transmitted together, so they are grouped in a single value: 'UE-AMBR UL/DL'.
- The values 'QCI_{HSS}', 'ARP_{HSS}', '$\text{APN-AMBR UL}_{\text{HSS}}$' and '$\text{APN-AMBR DL}_{\text{HSS}}$' are always transmitted together, so they can be grouped in a single value: '$\text{QoS Profile}_{\text{HSS}}$'.
- The values 'QCI_{SPR}', 'ARP_{SPR}', '$\text{APN-AMBR UL}_{\text{SPR}}$' and '$\text{APN-AMBR DL}_{\text{SPR}}$' are always transmitted together, so they can be grouped in a single value: '$\text{QoS Profile}_{\text{SPR}}$'.
- 'EPS Bearer ID' and 'E-RAB ID' correspond to the same value and their usage is the same. For these reasons, just one value will be considered for both: 'EPS Bearer ID'.
- The values 'DRB UL ID' and 'DRB DL ID' are always transmitted together, so they can be grouped in a single value: 'DRB ID UL/DL'.

A.2 Active_to_Idle Transition

- Step 1: UE Context Release Request (eNB → MME)

  The eNB informs the MME about the UE inactivity.

  Information carried:
  - eNB S1AP UE ID [21]
  - MME S1AP UE ID [21]

  Information destroyed:

  None

- Step 2: Release Access Bearers Request (MME → S-GW)

  The MME instructs the S-GW to release the resources for the downlink channel.

  Information carried:

  - EPS Bearer ID [15]

  Information destroyed:

  - S1 TEID DL (S-GW)

- Step 3: Release Access Bearers Response (MME ← S-GW)

  Acknowledgement of the S-GW to the MME: from now on, any downlink packet is buffered in the S-GW.

  Information carried:

  None

  Information destroyed:

  None

- Step 4: UE Context Release Command (eNB ← MME)

  The message instructs the eNB to release the context associated with the UE.

  Information carried:

  - eNB S1AP UE ID [21]
  - MME S1AP UE ID [21]

  Information destroyed:

  None

- Step 5: RRC Connection Release (UE ← eNB)

  The message releases the RRC connection between UE and eNB, deleting the UE context from the eNB.

  Information carried:

  - C-RNTI

  Information destroyed:

  - C-RNTI (UE)
  - ECGI (UE)
  - AS Security Algorithm (UE)
  - DRB ID UL/DL (UE)
  - C-RNTI (eNB)
  - ECGI (eNB)
  - TAI (eNB)
  - UE Security Algorithm (eNB)
  - AS Security Algorithm (eNB)
  - K_{eNB} (eNB)
  - EPS Bearer ID (eNB)
  - DRB ID UL/DL (eNB)
  - S1 TEID UL (eNB)

  - S1 TEID DL (eNB)
  - QCI_{PCC} (eNB)
  - ARP_{PCC} (eNB)
  - UE-AMBR UL (eNB)
  - UE-AMBR DL (eNB)

- Step 6: UE Context Release Complete (eNB → MME)

  The message acknowledges the successful deletion of the UE context.

  Information carried:

  - eNB S1AP UE ID [21]
  - MME S1AP UE ID [21]

  Information destroyed:

  - eNB S1AP UE ID (eNB)
  - MME S1AP UE ID (eNB)
  - eNB S1AP UE ID (MME)
  - MME S1AP UE ID (MME)
  - ECGI (MME)
  - S1 TEID DL (MME)

Please note that, at the end of the procedure, the ECGI value is still present in S-GW, P-GW and PCRF. To the best of our knowledge, there are no explicit requests sent to them in order to delete that value. We think that it will not be used till the next access to the network of the UE, which will trigger an overwriting of this information with the new value.

A.3 Idle_to_Active Transition (UE Triggered)

- Step 1: Random Access Preamble (UE → eNB)

  This message is sent by the UE to signal its will to join the network of the eNB.

  Information carried:

  - GUTI[1]

  Information created:

  - C-RNTI (eNB)

- Step 2: Random Access Response (UE ← eNB)

  This message is sent by the eNB to regulate the access to the wireless channel.

  Information carried:

  - C-RNTI

[1] A section of the GUTI, the S-TMSI, is used in place of a random value in order to access to the radio-level network. Practically, the eNB is not going to use the GUTI, it is used just to regulate the access to the shared media.

Information created:

None

- Step 3: RRC Connection Request (UE → eNB)
  The message creates a control plane channel between UE and eNB.
  Information carried:
  None
  Information created:
  None
- Step 4: RRC Connection Setup (UE ← eNB)
  The message regulates the parameters of the control plane channel.
  Information carried:
  None
  Information created:
  None
- Step 5: RRC Connection Setup Complete (UE → eNB)
  The message acknowledges the channel establishment.
  Information carried:
  - GUTI (relay)
  - KSI_{ASME} (relay)

  Information created:
  - eNB S1AP UE ID (eNB)
- Step 6: Initial UE Message (eNB → MME)
  The message creates a control plane channel between eNB and MME.
  Information carried:
  - GUTI
  - ECGI
  - TAI
  - KSI_{ASME}
  - eNB S1AP UE ID

  Information created:
  - MME S1AP UE ID (MME)
- Step 7: Initial Context Setup Request (eNB ← MME)
  The messages provides information to the eNB to rebuild the data plane channel.
  Information carried:
  - EPS Bearer ID

  - K_{eNB}
  - UE Security Algorithm
  - QCI_{PCC}
  - ARP_{PCC}
  - APN-AMBR UL_{PCC} (relay)
  - UE-AMBR UL
  - UE-AMBR DL
  - S1 TEID UL
  - MME S1AP UE ID

  Information created:

  - AS Security Algorithm (eNB)

- Step 8: AS Security Mode Command (UE ← eNB)

  The message establishes a secure communication between UE and eNB.

  Information carried:

  - AS Security Algorithm

  Information created:

  None

- Step 9: AS Security Mode Complete (UE → eNB)

  This message completes the security operations between UE and eNB.

  Information carried:

  None

  Information created:

  - DRB ID UL/DL (eNB)

- Step 10: RRC Connection Reconfiguration (UE ← eNB)

  The message creates the DRB bearer between eNB and UE.

  Information carried:

  - EPS Bearer ID
  - DRB ID UL/DL
  - QCI_{PCC}
  - APN-AMBR UL_{PCC}[2]

  Information created:

[2]Here we are supposing that MME sends QCI_{PCC} and APN-AMBR UL_{PCC} to the eNB, even if the UE should already know those values. While QCI_{PCC} has to be received by the eNB (it actually needs that value), the APN-AMBR UL_{PCC} is useful just to the UE. It seems that UE does not need QCI_{PCC} neither APN-AMBR UL_{PCC}, but the RRC Connection Reconfiguration message contains also some QoS values for the DRB. Our suspicion is that those possibly modified values are going to overwrite the ones into the UE.

  - S1 TEID DL (eNB)

- Step 11: Initial Context Setup Response (eNB → MME)

  The message informs the MME about the creation of the downlink endpoint.

  Information carried:

  - EPS Bearer ID
  - S1 TEID DL

  Information created:

  None

- Step 12: Modify Bearer Request (MME → S-GW)

  The message informs the S-GW about the downlink endpoint of the eNB.

  Information carried:

  - EPS Bearer ID
  - S1 TEID DL
  - ECGI [15]

  Information created:

  None

- Step 13: Modify Bearer Response (MME ← S-GW)

  The message acknowledges the reception of the downlink endpoint of the eNB by the S-GW.

  Information carried:

  None

  Information created:

  None

A.4 Idle_to_Active Transition (Network Triggered)

In this section, we present just the first four signals of the whole procedure, as the remaining signals are the same presented in Section A.3.

- Step 1: Downlink Data Notification (MME ← S-GW)

  The message is sent to notify the MME about the presence of incoming traffic for the UE.

  Information carried:

  - IMSI
  - EPS Bearer ID

  Information created:

  None

- Step 2: Downlink Data Notification Ack (MME → S-GW)

  The message acknowledges the request for paging.

  Information carried:

  – IMSI

  Information created:

  None

- Step 3: Paging (eNB ← MME)

  The message instructs the eNBs to forward the message in the TAs specified in the TAI List.

  Information carried:

  – IMSI
  – TAI List

  Information created:

  None

- Step 4: Paging (UE ← eNB)

  The message is sent to notify the UE to change from Idle state to Active state.

  Information carried:

  – IMSI
  – TAI List

  Information created:

  None

A.5 X2 Handover

- Step 1: Measurement Report (UE → SeNB)

  The message contains the information that allow the eNB to decide if an handover has to be performed or not.

  Information carried:

  – ECGI_{NEW} (relay)

  Information created:

  – $\text{K}_{\text{eNB NEW}}$ (SeNB)

  Information destroyed:

  None

- Step 2: Handover Request (SeNB → TeNB)

  The message asks to the TeNB if it can handle a new incoming UE.

  Information carried:

  - MME S1AP UE ID
  - ECGI_{NEW}
  - $\text{K}_{\text{eNB NEW}}$
  - UE Security Algorithm
  - EPS Bearer ID
  - S1 TEID UL
  - QCI_{PCC}
  - ARP_{PCC}
  - UE-AMBR UL/DL

  Information created:

  - DRB ID $\text{UL/DL}_{\text{NEW}}$ (TeNB)
  - $\text{C-RNTI}_{\text{NEW}}$ (TeNB)
  - X2 TEID DL (TeNB)
  - $\text{AS Security Algorithm}_{\text{NEW}}$ (TeNB)

  Information destroyed:

  None

- Step 3: Handover Request Ack (SeNB ← TeNB)

  The message acknowledges the handover request.

  Information carried:

  - EPS Bearer ID
  - X2 TEID DL
  - $\text{C-RNTI}_{\text{NEW}}$ (relay)
  - $\text{DRB ID UL/DL}_{\text{NEW}}$ (relay)
  - $\text{AS Security Algorithm}_{\text{NEW}}$ (relay)

  Information created:

  None

  Information destroyed:

  None

- Step 4: Handover Command (UE ← SeNB)

  The message instructs the UE to detach from the current base station and to attach to the new one.

  Information carried:

  - $\text{C-RNTI}_{\text{NEW}}$
  - $\text{DRB ID UL/DL}_{\text{NEW}}$
  - $\text{AS Security Algorithm}_{\text{NEW}}$

  Information created:

  - DL Count (SeNB)
  - UL Count (SeNB)

  Information destroyed:

  - C-RNTI (UE)
  - ECGI (UE)
  - AS Security Algorithm (UE)
  - DRB ID UL/DL (UE)

- Step 5: Sequence Number Status Transfer (SeNB → TeNB)

  The message carries the packet number from which the eNB should start to buffer downlink and uplink packets.

  Information carried:

  - DL Count
  - UL Count

  Information created:

  None

  Information destroyed:

  None

- Step 6: Handover Confirm (UE → TeNB)

  The message confirms the successful handover to the TeNB.

  Information carried:

  - C-RNTI$_{\text{NEW}}$

  Information created:

  - eNB S1AP UE ID$_{\text{NEW}}$ (TeNB)
  - S1 TEID DL$_{\text{NEW}}$ (TeNB)

  Information destroyed:

  None

- Step 7: Path Switch Request (TeNB → MME)

  The TeNB informs the MME about the new downlink endpoint.

  Information carried:

  - eNB S1AP UE ID$_{\text{NEW}}$
  - MME S1AP UE ID
  - ECGI$_{\text{NEW}}$
  - TAI
  - UE Security Algorithm
  - EPS Bearer ID

  - S1 TEID DL_{NEW}

  Information created:

  None

  Information destroyed:

  - eNB S1AP UE ID (MME)
  - ECGI (MME)
  - S1 TEID DL (MME)

- Step 8: Modify Bearer Request (MME → S-GW)
  The MME informs the S-GW about the new downlink endpoint.
  Information carried:

  - $ECGI_{NEW}$
  - TAI
  - EPS Bearer ID
  - S1 TEID DL_{NEW}

  Information created:

  None

  Information destroyed:

  - ECGI (S-GW)
  - S1 TEID DL (S-GW)

- Step 9: Modify Bearer Request (S-GW → P-GW)
  The S-GW forwards the location information to the P-GW.
  Information carried:

  - EPS Bearer ID
  - $ECGI_{NEW}$
  - TAI

  Information created:

  None

  Information destroyed:

  - ECGI (P-GW)

- Step 10: EPS Session Modification Notification (P-GW → PCRF)
  The P-GW forwards the information about the new location to the PCRF.
  Information carried:

  - IMSI
  - UE IP Address
  - $ECGI_{NEW}$

  - TAI

  Information created:

  None

  Information destroyed:

  - ECGI (PCRF)

- Step 11: EPS Session Modification Ack (P-GW ← PCRF)

  The PCRF acknowledges the reception of the updated location information.

  Information carried:

  None

  Information created:

  None

  Information destroyed:

  None

- Step 12: Modify Bearer Response (S-GW ← P-GW)

  The P-GW acknowledges the S-GW about the successful delivery of the location information.

  Information carried:

  - EPS Bearer ID

  Information created:

  None

  Information destroyed:

  None

- Step 13: Modify Bearer Response (MME ← S-GW)

  The S-GW acknowledges the MME about the successfully delivery of the location information.

  Information carried:

  - EPS Bearer ID

  Information created:

  None

  Information destroyed:

  None

- Step 14: Path Switch Request Ack (TeNB ← MME)

  The MME informs the TeNB that the downlink path has been successfully redirected.

  Information carried:

  - EPS Bearer ID

Information created:

None

Information destroyed:

None

- Step 15: UE Context Release (SeNB ← TeNB)

  The TeNB instructs the SeNB to release the context of the UE.

  Information carried:

  – X2 TEID DL

  Information created:

  None

  Information destroyed:

  – C-RNTI (SeNB)
  – eNB S1AP UE ID (SeNB)
  – MME S1AP UE ID (SeNB)
  – ECGI (SeNB)
  – K_{eNB} (SeNB)
  – AS Security Algorithm (SeNB)
  – EPS Bearer ID (SeNB)
  – DRB ID UL/DL (SeNB)
  – S1 TEID UL (SeNB)
  – S1 TEID DL (SeNB)
  – QCI_{PCC} (SeNB)
  – ARP_{PCC} (SeNB)
  – UE-AMBR UL/DL (SeNB)
  – UL Count (SeNB)
  – DL Count (SeNB)
  – X2 TEID DL (SeNB)
  – X2 TEID DL (TeNB)

A.6 S1 Handover

- Step 1: Measurement Report (UE → SeNB)

  The message contains the information that allow the eNB to decide if a handover has to be performed or not.

  Information carried:

  – $ECGI_{NEW}$ (relay)

Information created:

None

Information destroyed:

None

- Step 2: Handover Required (SeNB → MME)
  The message is sent towards the MME in order to ask for a S1 handover.
  Information carried:
  - MME S1AP UE ID [21]
  - eNB S1AP UE ID [21]
  - ECGI_{NEW}

  Information created:

  None

  Information destroyed:
  - ECGI (MME)
- Step 3: Handover Request (TeNB ← MME)
  The MME performs the handover request to the TeNB in place of the SeNB.
  Information carried:
  - MME S1AP UE ID
  - UE-AMBR UL
  - UE-AMBR DL
  - EPS Bearer ID
  - QCI_{PCC}
  - ARP_{PCC}
  - S1 TEID UL
  - UE Security Algorithm
  - ECGI_{NEW}

  Information created:
  - AS Security $\text{Algorithm}_{\text{NEW}}$ (TeNB)
  - $\text{K}_{\text{eNB NEW}}$ (TeNB)
  - eNB S1AP UE ID_{NEW} (TeNB)
  - S1 TEID DL_{NEW} (TeNB)
  - S1 eNB TEID DL (TeNB)
  - C-RNTI_{NEW} (TeNB)
  - DRB ID UL/DL_{NEW} (TeNB)

  Information destroyed:

  None

- Step 4: Handover Request Ack (TeNB → MME)

  The TeNB confirms to the MME its availability to handle the new incoming UE.

  Information carried:

  - EPS Bearer ID
  - eNB S1AP UE ID_{NEW}
  - S1 TEID DL_{NEW}
  - S1 eNB TEID DL (relay)
  - AS Security $Algorithm_{NEW}$ (relay)
  - C-$RNTI_{NEW}$ (relay)
  - DRB ID UL/DL_{NEW} (relay)

  Information created:

  None

  Information destroyed:

  - S1 TEID DL (MME)

- Step 5: Create Indirect Data Forwarding Tunnel Request (MME → S-GW)

  The MME asks the S-GW to create the indirect tunnel between the two eNBs in order to let the buffered packets reaching the TeNB.

  Information carried:

  - EPS Bearer ID
  - S1 eNB TEID DL

  Information created:

  - S1 S-GW TEID UL (S-GW)

  Information destroyed:

  None

- Step 6: Create Indirect Data Forwarding Tunnel Response (MME ← S-GW)

  The S-GW answers back to the MME with the tunnel endpoint that has to be used by the SeNB.

  Information carried:

  - EPS Bearer ID
  - S1 S-GW TEID UL (relay)

  Information created:

  None

  Information destroyed:

  None

- Step 7: Handover Command (SeNB ← MME)

  The message informs the SeNB that everything needed for the handover has been prepared.

  Information carried:

  – S1 S-GW TEID UL
  – C-RNTI$_{\text{NEW}}$ (relay)
  – DRB ID UL/DL$_{\text{NEW}}$ (relay)
  – AS Security Algorithm$_{\text{NEW}}$ (relay)

  Information created:

  None

  Information destroyed:

  None

- Step 8: Handover Command (UE ← SeNB)

  The message instructs the UE to perform the handover.

  Information carried:

  – C-RNTI$_{\text{NEW}}$
  – DRB ID UL/DL$_{\text{NEW}}$
  – AS Security Algorithm$_{\text{NEW}}$

  Information created:

  – UL Count (SeNB)
  – DL Count (SeNB)

  Information destroyed:

  – C-RNTI (UE)
  – ECGI (UE)
  – AS Security Algorithm (UE)
  – DRB ID UL/DL (UE)

- Step 9: eNB Status Transfer (SeNB → MME)

  The message informs from which packet number the eNB should start buffering downlink and uplink packets.

  Information carried:

  – DL Count (relay)
  – UL Count (relay)

  Information created:

  None

  Information destroyed:

  None

- Step 10: MME Status Transfer (TeNB ← MME)

  The message informs from which packet number the eNB should start buffering downlink and uplink packets.

  Information carried:

  - DL Count
  - UL Count

  Information created:

  None

  Information destroyed:

  None

- Step 11: Handover Confirm (UE → TeNB)

  The message confirms the successful handover to the TeNB.

  Information carried:

  - $\text{C-RNTI}_{\text{NEW}}$
  - ECGI_{NEW}

  Information created:

  None

  Information destroyed:

  None

- Step 12: Handover Notify (TeNB → MME)

  The TeNB informs the MME that the handover has been successfully completed.

  Information carried:

  - ECGI_{NEW}
  - TAI

  Information created:

  None

  Information destroyed:

  None

- Step 13: Modify Bearer Request (MME → S-GW)

  The MME informs the S-GW about the new S1 TEID DL in the TeNB.

  Information carried:

  - EPS Bearer ID
  - ECGI_{NEW}
  - TAI
  - $\text{S1 TEID DL}_{\text{NEW}}$

Information created:

None

Information destroyed:

- ECGI (S-GW)
- S1 TEID DL (S-GW)

- Step 14: Modify Bearer Request (S-GW → P-GW)
  The S-GW forwards the location information to the P-GW.
  Information carried:
  - EPS Bearer ID
  - ECGI_{NEW}
  - TAI

  Information created:

  None

  Information destroyed:
  - ECGI (P-GW)
- Step 15: EPS Session Modification Notification (P-GW → PCRF)
  The P-GW forwards the information about the new location to the PCRF.
  Information carried:
  - IMSI
  - UE IP Address
  - ECGI_{NEW}
  - TAI

  Information created:

  None

  Information destroyed:
  - ECGI (PCRF)
- Step 16: EPS Session Modification Ack (P-GW ← PCRF)
  The PCRF acknowledges the reception of the updated location information.
  Information carried:

  None

  Information created:

  None

  Information destroyed:

  None

- Step 17: Modify Bearer Response (S-GW ← P-GW)

  The P-GW acknowledges the S-GW about the successfully delivery of the location information.

  Information carried:

  – EPS Bearer ID

  Information created:

  None

  Information destroyed:

  None

- Step 18: Modify Bearer Response (MME ← S-GW)

  The S-GW acknowledges the MME about the successfully delivery of the location information.

  Information carried:

  – EPS Bearer ID

  Information created:

  None

  Information destroyed:

  – S1 eNB TEID DL (TeNB)[3]

- Step 19: UE Context Release Command (SeNB ← MME)

  The MME instructs the SeNB to release the UE context.

  Information carried:

  None

  Information created:

  None

  Information destroyed:

  – C-RNTI (SeNB)
  – eNB S1AP UE ID (SeNB)
  – MME S1AP UE ID (SeNB)
  – ECGI (SeNB)
  – K_{eNB} (SeNB)
  – UE Security Algorithm (SeNB)
  – AS Security Algorithm (SeNB)
  – EPS Bearer ID (SeNB)
  – DRB ID UL/DL (SeNB)

[3]This information is destroyed because of the "End marker" sent by the S-GW to the TeNB through the indirect tunnel, which signals that no more packets will be forwarded through the tunnel any more.

  - S1 TEID UL (SeNB)
  - S1 TEID DL (SeNB)
  - QCI_{PCC} (SeNB)
  - ARP_{PCC} (SeNB)
  - UE-AMBR UL/DL (SeNB)
  - S1 S-GW TEID UL (SeNB)
  - UL Count (SeNB)
  - DL Count (SeNB)

- Step 20: UE Context Release Complete (SeNB → MME)

  The SeNB acknowledges the deletion of the UE context.

  Information carried:

  None

  Information created:

  None

  Information destroyed:

  None

- Step 21: Delete Indirect Data Forwarding Tunnel Request (MME → S-GW)

  The MME instructs the S-GW to release the resources for the indirect tunnel between the two eNBs.

  Information carried:

  None

  Information created:

  None

  Information destroyed:

  - S1 eNB TEID DL (S-GW)
  - S1 S-GW TEID UL (S-GW)

- Step 22: Delete Indirect Data Forwarding Tunnel Response (MME ← S-GW)

  The S-GW acknowledges the MME of the successful deletion of the indirect tunnel information.

  Information carried:

  None

  Information created:

  None

  Information destroyed:

  None

Appendix B

CoNEXT'16 submission

We have used the ideas that came out from our work to write a 6-pages paper, which has been submitted to the 12th International Conference on emerging Networking EXperiments and Technologies (CoNEXT), which will take place in Irvine, California, in December 2016. The paper is focused on the flexibility injected in the LTE architecture by our modularization work, especially the first refactoring step. In fact, the outcome of the third refactoring step is considered just as one of the possible implementations of the modularized architecture coming out from the first refactoring step. In this context, the avoidance of signaling storms is just one of the many verticals for which the LTE network can be customized. The submitted article follows.

A Refactoring Approach for Optimizing Mobile Networks

Matteo Pozza[†‡], Ashwin Rao[†], Armir Bujari[‡],
Hannu Flinck[*], Claudio E. Palazzi[‡], and Sasu Tarkoma[†]
[†]University of Helsinki, [‡]University of Padua, [*]Nokia Bell Labs

ABSTRACT

Mobile networks are expected to serve a wide range of verticals such as Internet of Things, however the Long Term Evolution (LTE) network is optimized for basic mobile operator services and offers limited services to other verticals. Furthermore, the network functions serving an LTE network are largely implemented as dedicated single function devices that exchange a large number of signaling messages for replicating the state of the mobile devices using the LTE network. Modularizing these network functions would enable a refactoring of the LTE network, allowing operators to compose networks that adapt and evolve with the influx of verticals. In this article, we present a new approach for refactoring the network functions serving LTE networks. In particular, we show that it is possible to separate and modularize the control and data planes of LTE networks without modifying the mobile devices. With the help of three use-cases, we show that our refactoring approach can be leveraged to compose a modular mobile network optimized for the verticals using its services. We also show that deploying network functions at the edge significantly reduces the signals exchanged within a mobile network.

1. INTRODUCTION

The next generation mobile networks are expected to integrate and serve a wide range of verticals such as Internet of Things. Each of these verticals is expected to arrive with a unique set of requirements from the control plane and data plane. For example, critical communication such as autonomous driving has different control plane and data plane requirements compared to wireless sensors that are expected to be deployed in a house.

This influx of verticals imposes a serious scalability challenge to the Long Term Evolution (LTE) network [2]. The LTE network is optimized for mobile operator services, as it is the outcome of satisfying the requirement of a high-speed data network while being compatible with previous generation mobile networks. The network functions serving an LTE network are largely implemented as dedicated single function devices, some of which serve the control plane and the data plane [3, 14]. Moreover, these single function devices exchange a large number of messages for replicating the state of mobile devices using the network [16, 19]. These network functions also offer limited customization options, resulting in an inflexible network [2, 21]. The current LTE networks are therefore unable to adapt and evolve with the influx of verticals.

Modularizing these network functions will enable operators to refactor the network for satisfying the requirements of the verticals using its services. In this article, we detail our approach for refactoring the network functions serving LTE networks. We also present three use cases to exemplify the benefits of our approach.

Our two key contributions are as follows.

1. We analyze and abstract the roles of the various network functions serving LTE networks, and we also analyze the messages they exchange with each other. This gives us insights on the avenues for optimizing mobile networks to satisfy the requirements of the verticals using their services.
2. We show that it is indeed possible to separate and modularize the control and data plane of the network functions without modifying the mobile devices. This enables us to explore the impact of refactoring the network functions. For example, we show that moving the functionality to the network edge reduces significantly the signaling messages exchanged by the network functions.

We were heavily inspired by the seminal works of Li *et al.* [14] and Gudipati *et al.* [9], which detail the benefits of bringing Software Defined Networking (SDN) to cellular networks. We also leverage the insights of previous works that propose a logically centralized control plane for mobile networks [7, 13, 15, 17]. Similarly, Hampel *et al.* [10] have also explored the benefits of splitting the control plane and data plane of the network functions. Our use cases are based on the insights from Moradi *et al.* [17] and Jin *et al.* [12], which advocate splitting the control plane and moving some of its components to the base stations.

The rest of the paper is organized as follows. In §2 we discuss the inflexibility of the network functions in LTE networks. Then we discuss our approach to abstract

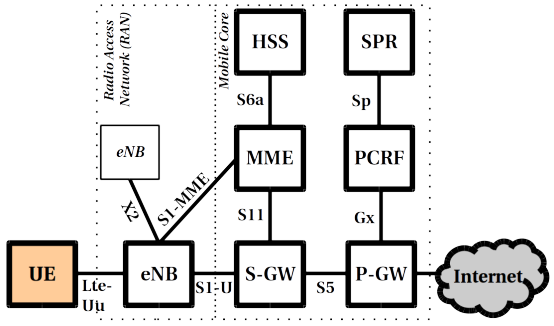


Figure 1: **Key network functions and interfaces in an LTE Network.** *An LTE network consists of the Radio Access Network (RAN) and the Mobile Core.*

Procedure	Number of signals per interface									Total no. of signals
	LTE-Uu	*X2*	*S1-U*	*S1-MME*	*S11*	*S6a*	*S5*	*Gx*	*Sp*	
IA	13	0	0	8	4	4	2	2	2	**35**
AtI	1	0	0	3	2	0	0	0	0	**6**
ItA (UE)	8	0	0	3	2	0	0	0	0	**13**
ItA (Net)	9	0	0	4	4	0	0	0	0	**17**
X2H	3	4	0	2	2	0	2	2	0	**15**
S1H	3	0	0	9	6	0	2	2	0	**22**

Table 1: **Number of signals exchanged in LTE networks during the Initial Attach (IA), Active to Idle (AtI), Idle to Active (ItA), and Handover procedures.** *For ItA, we consider both sub-types: UE driven (UE) and Network driven (Net). We also consider two types of handovers: X2 handover, which uses the X2 link, and S1 handover, performed when the X2 link is missing. Each procedure requires a large number of messages (signals) exchanged between the various network functions serving an LTE network.*

and modularize the network functions in §3, followed by some use cases in §4 and a discussion in §5.

2. BACKGROUND AND MOTIVATION

In this section, we first provide an overview of the key network functions serving LTE networks. Then, we highlight the shortcomings of the LTE networks.

2.1 Overview of the LTE Architecture

As shown in Figure 1, current LTE networks consist of two sub networks: the Radio Access Network (RAN) and the Mobile Core [3].

The key network function in the RAN is the Evolved Node B (eNB). The eNBs use their Lte-Uu interface to communicate with mobile devices, henceforth referred to as User Equipment (UE). Furthermore, eNBs use their X2 interface to communicate with some of the other eNBs in the LTE network. This X2 interface is useful during handovers, and network operators determine the set of eNBs to which an eNB communicates using its X2 interface. An eNB is also connected to two network functions of the Mobile Core: the Serving Gateway (S-GW) via the S1-U interface, and the Mobile Management Entity (MME) via the S1-MME interface.

In Figure 1, we present the key network functions in the Mobile Core and the interfaces they use to communicate with each other. For example, the S-GW uses the S11 interface to communicate with the MME. The key roles of the MME include a) ensuring smooth handovers, and b) retrieving customer-specific values from the Home Subscriber Server (HSS), a database storing the users' subscription information. The S-GW acts as a mobility anchor and it is typically linked to more than one base station. It is also linked to the Packet Data Network Gateway (P-GW), the gateway to the external networks. The P-GW also assigns the IP addresses to the UEs and enforces the QoS rules received by the Policy and Charging Rules Function (PCRF). The PCRF uses the values retrieved by the Subscriber Profile Repository (SPR) in turn to generate the rules fed to the P-GW.

A UE served by these network functions can change its status for various reasons. For example, a user can power on or off a UE, or the UE may sleep and go idle to save energy, or the user can move from one location to another. When a UE changes its status, the UE and the network functions serving the UE undergo a series of actions called *procedures*. Furthermore, the network functions internally exchange a large number of messages called *signals* to ensure a consistent state of the UEs they serve. We now leverage this background to discuss the shortcomings of the LTE architecture.

2.2 Shortcomings of the LTE Architecture

Current LTE networks are inflexible because their key network functions are implemented as dedicated single function devices [4]. Furthermore, the eNB, S-GW, and P-GW, serve both the control plane and the data plane [14, 21]. Because of their convoluted roles and lack of customization options, these single function devices offer limited support to mobile operators for optimizations such as power saving and handling flash crowds. This is a serious shortcoming given the wide range of verticals expected to use future mobile networks [2].

For example, as shown in Table 1, the network functions exchange a large number of signals to replicate the state of UEs they serve [19, 6]. In this article, we restrict our analysis to four frequently used procedures: a) *Initial Attach* (IA) which is executed when the UE connects to an LTE network, b) *Active to Idle* (AtI) which is executed when the UE goes idle, c) *Idle to Active* (ItA) which is executed when an idle UE becomes

active, and d) *Handover* which is executed during UE mobility. We focus on these procedures because we believe that they will be required by a significant number of verticals. Furthermore, the UE state transitions and the handover procedures are some of the key sources of the signaling load in the LTE networks [5], and the Initial Attach requires one of the highest number of signals per procedure. In fact, during the Initial Attach procedure, the UE sends 13 signals to the LTE network and the network functions exchange 22 signals among themselves to complete the procedure.

This convoluted implementation of the network functions makes the current LTE networks inflexible and unable to adapt with the influx of demands from mobile networks. For example, popular applications for mobile phones require periodic heartbeat messages, and such short data sessions are one the primary causes of the signaling storms in LTE networks [19]. In the following, we present our approach to address this inflexibility.

3. ABSTRACT AND MODULARIZE THE NETWORK FUNCTIONS

Our goal is to transform an LTE network into a network that can evolve and adapt to serve the wide range of services expected from the next generation (5G) mobile networks. Our assumptions and guidelines for this transformation are as follows.

1. **Unmodified mobile devices.** A network will be useful to the users when existing mobile devices can be used as-is. Therefore, we do not modify any procedures at the UEs, including the security and authentication procedures. This requirement also provides a starting point to gain insights for a clean slate approach to optimize mobile networks.
2. **Scale to new verticals.** Future mobile networks are expecting an influx of verticals whose control plane and data plane scale very differently. For example, a network optimized for entertainment services has different control plane and data plane requirements compared to a network setup during natural disasters. Therefore, scaling to these verticals inherently requires splitting the control and data plane. On these lines, we aim to make the data plane as simple as possible, limiting its functions to the enforcement of the rules received and to the notification of network events.

We achieve our goal by first abstracting the roles of the network function in §3.1, and then leveraging these abstractions to identify the modules in §3.2.

3.1 Abstract the Roles of Network Functions

Abstractions distill the underlying simplicity of complex communication systems and provide an ideal vantage point to study them [22]. The Software-Defined

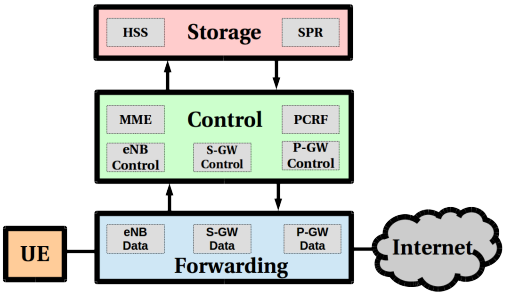


Figure 2: **Three-layers abstraction.** *An LTE network can be abstracted into a Storage layer, a Control layer, and a Forwarding layer.*

Networking paradigm provides us two key abstractions for any communication systems: the control plane and the data plane. The MME and the PCRF serve only the control plane of LTE networks, while the eNB, S-GW, and P-GW serve both the control plane and the data plane. However, these two planes are not exhaustive in the context of LTE networks. In particular, the HSS and the SPR store the subscription information used to authenticate UEs and to generate the policies enforced by the gateways. While HSS and SPR can be viewed to be part of the control plane, we argue that these network functions are databases used by the control plane.

As shown in Figure 2, we use three layers to abstract an LTE network: a) the *Storage* layer for persistent data and database-like services, b) the *Control* layer for the management of the forwarding elements, and c) the *Forwarding* layer for handling data flows according to the rules imposed by the *Control* layer.

The *Forwarding* layer acts as the data bridge between the UE and the Internet. It also forwards the control plane traffic from the UE to the *Control* layer (*e.g.*, signals during authentication). The *Control* layer takes decisions based on the statistics coming from the data plane (*e.g.*, bandwidth, queue length in the gateways, *etc.*) and the policies generated using the information in the *Storage* layer. Consequently, it instructs the forwarding devices on how to serve the packets.

3.2 Leverage Abstractions to Identify Modules

The eNB, the S-GW, and the P-GW serve the control plane and the data plane. Therefore, we split each one of these network functions into two modules, one for the *Forwarding* layer and the other for the *Control* layer. We also explicitly add a communication interface between these two modules. As shown in Figure 3, we split the eNB into two modules and we add an interface named M1-10 between these modules. The MME and the PCRF are modules in the *Control* layer while the HSS and SPR are modules in the *Storage* layer.

As shown in Table 2, this separation of modules adds

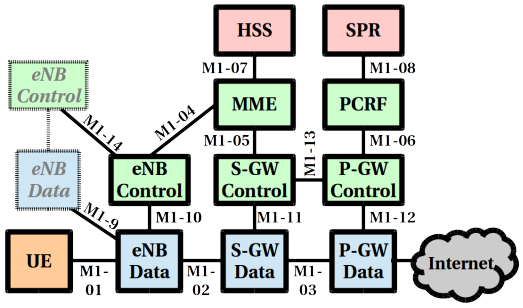


Figure 3: **Modularized network functions.** *The eNB, S-GW, and P-GW are split as Control layer and Forwarding layer modules with interfaces added to connect these modules. This step allows us to study the interaction between the modules and explore the impact of combining these modules.*

	Number of signals per interface													
Proc.	*M1-01*	*M1-02*	*M1-03*	*M1-04*	*M1-05*	*M1-06*	*M1-07*	*M1-08*	*M1-09*	*M1-10*	*M1-11*	*M1-12*	*M1-13*	*M1-14*	**Tot.**
IA	13	0	0	8	4	2	4	2	0	14	6	2	2	0	**57**
AtI	1	0	0	3	2	0	0	0	0	3	2	0	0	0	**11**
ItA(UE)	8	0	0	3	2	0	0	0	0	9	2	0	0	0	**24**
ItA(Net)	9	0	0	4	4	0	0	0	0	10	3	0	0	0	**30**
X2H	3	0	0	2	2	2	0	0	0	14	2	0	2	4	**31**
S1H	3	0	0	9	6	2	0	0	0	13	6	0	2	0	**41**

tabcolsep

Table 2: **Number of signals exchanged during the Initial Attach (IA), Active to Idle (AtI), Idle to Active (ItA), and Handover procedures when using modules.** *The number of signals indicate the extent of interaction between the modules.*

new interfaces that connect the modules and also inflates the number of signals exchanged during each procedure. This is particularly evident during initial attach procedure: while the number of signals between the UE and the eNB Data is the same as in existing LTE networks (13 signals), the total number of signals exchanged during the initial attach procedure increases to 57.[1] This step helps us study the interaction between the modules of the control plane and also identify avenues for optimization. For example, in Table 2, across all the procedures we consider, we observe that the number of signals exchanged over the M1-10 interface (between the eNB Control and Data modules) is more than the number of signals over the M1-11 interface (between the S-GW Control and Data modules) and the M1-12 interface (between the P-GW Control and Data modules). This implies that it is beneficial to keep the control and data plane of the RAN physically close to each other. Furthermore, one can also consider moving the control plane of the Mobile Core close to the RAN.

[1]The details of 57 signals are available in `https://drive.google.com/open?id=0B49I92-bJ59KdTg3SWRrelprWXM`. We plan to release a technical report detailing these signals.

3.3 Discussion

In this section, we modularize the network functions of LTE networks and we show that this can be achieved without modifying the mobile devices. This can be achieved in practice, and our code to split the control and data plane of the gateways [20] has been used for testing dynamic switching of tunnels used by the network functions [11]. Furthermore, Hampel *et al.* [10] have also split the gateways of mobile networks and explored the benefits of bringing SDN to mobile networks.

We also provide a detailed analysis about how the split affects the signals exchanged between the resulting modules. A key benefit of this work is that it enables us to explore different approaches to refactor the LTE network. For example, one can instantiate the modules of the control plane in a single physical device to reduce the number of signals exchanged between the physical devices serving the network. One can also explore the impact of putting the eNB control in the Mobile Core in order to keep the RAN composed by cheap and interchangeable base stations. We explore the impact of three such possibilities in the next section.

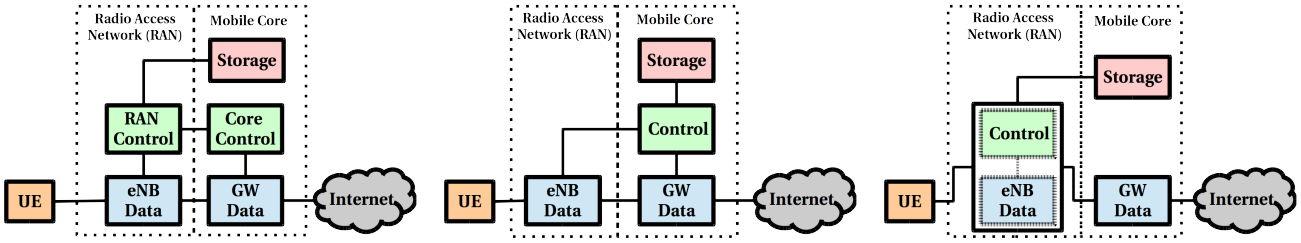
4. REFACTORING THE NETWORK FUNCTIONS

We now present three examples of refactoring the network functions. In each example, we refactor the network functions by coalescing in different ways the modules identified in §3.2. *Coalescing in this context implies running the modules in the same physical machine.* The coalesced modules internally exchange signals with each other but these signals do not leave the physical machine, resulting in a decrease in the number of signals traversing the interfaces connecting these physical machines. We use this exercise to explore the impact of composing network functions in specific ways.

4.1 Description of Examples

In the following three examples we coalesce the modules presented in the previous section. The modules in the *Forwarding* layer—the *eNB Data*, the *S-GW Data*, and the *P-GW Data*—can be conceptualized as SDN switches and routers which can be dynamically programmed by the *Control* layer. Similarly, we can consider the modules of the *Control* layer as software modules which can be coalesced and instantiated as virtual network functions [8]. Finally, the modules of the *Storage* layer can be coalesced to provide the database-like services required by the LTE network.

4.1.1 Split RAN & Core



(a) Split RAN & Core. (b) Thin Edge. (c) Intelligent Edge.

Figure 4: **Examples of refactoring LTE networks.** *The Split RAN & Core satisfies the requirement of having a control entity in the RAN and the Mobile Core while reducing the signals required to connect new devices to the mobile network. The Thin Edge satisfies the requirement of having all the control logic in the Mobile Core. The Intelligent Edge satisfies the requirement of having all the control logic in one physical box in the RAN.*

This example satisfies the requirement of having a control entity in the RAN and the Mobile Core while reducing the signals required for connecting new devices to the network, *i.e.*, reducing the total number of signals required to complete the Initial Attach procedure.

For this example, we coalesce the *Control* layer modules in two entities: the RAN Control and the Core Control. The RAN Control is the product of coalescing the eNB Control, the MME, and the PCRF, *i.e.*, these modules are instantiated on the same physical machine in the RAN; having the MME and PCRF in the same physical machine requires only one interface on the *Storage* device. Similarly, the Core Control is the outcome of coalescing the S-GW Control and P-GW Control, and the GW Data is an outcome of coalescing the S-GW Data and P-GW Data. We would like to point out that our approach is not completely new. In many LTE networks the S-GW and P-GW are largely implemented as a single box [4]; this example can also be seen as a step in using a legacy S-GW and P-GW in the Mobile Core while moving some of its control logic to the RAN. In this example, most of the decisions are taken in the RAN Control while the Core Control is only responsible for managing the *Forwarding* layer of the Mobile Core.

As presented in Table 3, we observe that the number of signals exchanged between the entities of this network during the Initial Attach decreases to 26.[2] The main cause for this decrease is the generation of the policy rules in the RAN. However, on the down side we observe that such a network requires more signals compared to existing LTE networks during all the other procedures. These increases come from the indirection between the RAN Control and the GW Data: in the LTE architecture, the MME communicates directly with the S-GW, while here we have the Core Control in between.

4.1.2 Thin Edge

This example is used to satisfy the requirement of having minimal resources in the RAN and having all the control and storage logic in the Mobile Core. As shown in Figure 4(b), in this example all the modules in the *Control* layer are coalesced and instantiated on a single physical server in the Mobile Core. The eNB Data serves the data plane of the RAN, while the GW Data serves the data plane of the Mobile Core.

As shown in Table 3, this refactoring and coalescing of modules results in fewer signals compared to existing LTE networks. For example, during the Initial Attach a total of 24 signals are exchanged between the various components; of these 24 signals, 13 signals involve the UE because the examples assume that the UE is not modified. The merging of the two control entities of the previous example is the primary reason behind the decrease in the number of signals. Nevertheless, this example results in an increase in the signals when handling an equivalent of X2 handovers.

	Total no. of signals per procedure					
Implementation	*IA*	*AtI*	*ItA (UE)*	*ItA (Net)*	*X2H*	*S1H*
LTE (Baseline)	35	6	13	17	15	22
Split RAN & Core	26	8	15	19	20	20
Thin Edge	24	6	13	16	16	16
Intelligent Edge	17	3	10	12	12	12

Table 3: **Total number of signals per procedure for each example**. *We present the signals observed by existing LTE networks as a base line for comparison.*

4.1.3 Intelligent Edge

This example is used to satisfy the requirement of having all the control logic in the RAN but having a storage in the Mobile Core. Furthermore, as shown in Figure 4(c), the modules in a control plane are instantiated as virtual machines in a device running the eNB Data. This is done to make use the of the free computation power which is available in the base stations. For example, Yousaf *et al.* [23] point out that up to 80% of the processing capacity of base stations is not used.

As shown in Table 3, this refactoring and coalescing

[2]The details of 26 signals are available in `https://drive.google.com/open?id=0B49I92-bJ59KUHFGZEpkVk9kTGM`.

	Frequency of procedures					
Dataset	*IA*	*AtI*	*ItA (UE)*	*ItA (Net)*	*X2H*	*S1H*
Metsälä *et al.* [16]	0.5	34	19	15	8	0.2

(a) Frequency of procedures.

	Frequency of signals					
Implementation	*IA*	*AtI*	*ItA (UE)*	*ItA (Net)*	*X2H*	*S1H*
LTE (Baseline)	17.5	204	247	255	120	4.4
Split RAN & Core	13	272	285	285	160	4
Thin Edge	12	204	247	240	128	3.2
Intelligent Edge	8.5	102	190	180	96	2.4

(b) Frequency of signals considering Metsälä *et al.* [16].

Table 4: **Impact of refactoring.** *We use the dataset of Metsälä et al. to quantify the impact of the different examples on the signaling load in the network, i.e., number of signals per busy hour per subscriber per base station for a given procedure.*

of modules results in fewer signals compared to existing LTE networks, and it also outperforms the example of the Thin Edge. For example, during the Initial Attach a total of 17 signals are exchanged between the various components; of these 17 signals, 13 signals involve the UE because our examples assume that the UE is not modified. The main reason behind this saving is the merging of control plane and eNB data in the same physical device. In fact, LTE networks require UEs to interact with the MME for authentication and exchanging security parameters, and the eNB acts as a relay for these messages. Coalescing the MME with the eNB Control and eNB Data decreases the number signals exchanged. We would like to point out that shifting the entire control logic to RAN can pose administrative and domain issues for some operators. However, these issues are beyond the scope of this paper. Our objective for presenting these example is to highlight the benefits of modularizing and refactoring the network functions.

4.2 Analysis

We now compare the signaling load generated when these approaches are deployed in an existing LTE network. For this comparison, we combine our results from Table 3 with real frequencies of the considered procedures. Metsälä *et al.* [16] provide data for the control plane traffic in an LTE network. We use a subset of this data, and in Table 4(a), we present the frequency of procedures per busy hour per subscriber per base station. For example, in their network there were an average of 0.5 initial attach procedures triggered by each subscriber per busy hour per base station. Please note that LTE networks in different geographical regions will have different values for the observed frequencies.

We observe that, for the given dataset, the Intelligent Edge approach outperforms the other approaches. Furthermore, this network has a small number of Initial Attach and a small number of S1 handovers compared to the X2 Handovers. As a consequence, the Split RAN & Core approach is not suitable for this network. Nevertheless, the Split RAN & Core approach is beneficial in networks with a high frequency of Initial Attach events. Finally, we can see the Thin Edge approach performs poorly only when handling the equivalent of X2 handovers. Therefore, this approach is beneficial for network deployments which do require base stations to be directly connected to each other. The objective of this simple exercise was to highlight how the network load affects the performance of the discussed use cases.

5. CONCLUSION AND FUTURE WORK

In this article, we abstract the roles of the key network functions serving LTE networks. This empowers us with a vantage point to modularize the network functions. These modules can be leveraged in turn to refactor the LTE network in order to serve specific deployment scenarios, some of which are discussed in §4.

Clean slate approach. In this article we assume that the UE cannot be modified, and optimizations, if any, are limited to the network functions serving the UE. While this offers a nice starting point to explore the extent to which existing devices can be used in future mobile networks, it also gives a vantage point to explore a clean slate approach. For example, in our Intelligent Edge use case, the UE exchanges 17 signals during the Initial Attach, and 13 of them are exchanged between the network function in the RAN and the UE. Clearly, the total number of signals can be further reduced if the refactoring approach is extended to the UE. We plan to extend this work to explore similar scenarios.

Network in a Box. We believe that the best way of maximizing modularity is by encapsulating modules of network functions in a single physical device, henceforth referred to as a *Box*. Each module in this *Box* can be enabled or disabled on demand, and every *Box* can communicate with other *Boxes*. Note that this *Box* does not contain hardware modules for the *Forwarding* layer but it can include the option to plug such hardware modules [1, 18]. Furthermore, such a *Box* can contain *shim* layers to communicate with legacy devices and network functions from existing LTE networks. A network operator can customize these *Boxes* according to the demands from the network. For example, a single *Box* can provide a complete ready-to-use cellular network, a common use case in natural disasters; alternatively, an operator could be interested to dynamically add cellular infrastructure in specific locations to handle crowds in case of sport events and music festivals.

To conclude, we believe that our work provides a key building block for composing modular mobile networks which can adapt to serve the influx verticals being in-

tegrated into mobile networks.

6. REFERENCES

[1] Endaga - community cellular networks. https://www.endaga.com/.

[2] Taking 5G to other industries - a view from the top. http://networks.nokia.com/news-events/insight-newsletter/articles/taking-5g-to-other-industries-a-view- from-the-top, 2015.

[3] 3GPP. General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access. TS 23.401, 3rd Generation Partnership Project (3GPP), Dec. 2015.

[4] Alcatel-Lucent. 7750 Service Router - Mobile Gateway. https://www.alcatel-lucent.com/products/7750-service-router-mobile-gateway.

[5] Alcatel-Lucent. Managing the signaling traffic in packet core. http://resources.alcatel-lucent.com/asset/155160.

[6] Banerjee, A., Mahindra, R., Sundaresan, K., Kasera, S., der Merwe, J. V., and Rangarajan, S. Scaling the LTE Control-Plane for Future Mobile Access. In *Proceedings of ACM CoNEXT* (2015).

[7] Basta, A., Kellerer, W., Hoffmann, M., Hoffmann, K., and Schmidt, E.-D. A Virtual SDN-Enabled LTE EPC Architecture: A Case Study for S-/P-Gateways Functions. In *Proc. of IEEE SDN for Future Networks and Services* (2013), pp. 1–7.

[8] European Telecommunications Standards Institute. Network Functions Virtualization - Introductory White Paper. https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf, 2014.

[9] Gudipati, A., Perry, D., Li, L. E., and Katti, S. Softran: Software defined radio access network. In *Proceedings of ACM HotSDN* (2013), pp. 25–30.

[10] Hampel, G., Steiner, M., and Bu, T. Applying Software-Defined Networking to the telecom domain. In *IEEE INFOCOM Workshops* (2013), pp. 133–138.

[11] Heinonen, J., Partti, T., Kallio, M., Lappalainen, K., Flinck, H., and Hillo, J. Dynamic Tunnel Switching for SDN-based Cellular Core Networks. In *Proceedings of the ACM AllThingsCellular Workshop* (2014), pp. 27–32.

[12] Jin, X., Li, L. E., Vanbever, L., and Rexford, J. SoftCell: Scalable and Flexible Cellular Core Network Architecture. In *Proc. of ACM CoNEXT* (2013), pp. 163–174.

[13] Kempf, J., Johansson, B., Pettersson, S., Luning, H., and Nilsson, T. Moving the mobile Evolved Packet Core to the cloud. In *Proc. of IEEE WiMob* (2012), pp. 784–791.

[14] Li, L. E., Mao, Z. M., and Rexford, J. Toward Software-Defined Cellular Networks. In *Proceedings of the European Workshop on Software Defined Networking* (2012), pp. 7–12.

[15] Lindholm, H., Osmani, L., Flinck, H., Tarkoma, S., and Rao, A. State Space Analysis to Refactor the Mobile Core. In *Proceedings of ACM AllThingsCellular Workshop* (2015), pp. 31–36.

[16] Metsälä, E. M., and Salmelin, J. *LTE Backhaul - Planning and Optimization*. Wiley, 2015.

[17] Moradi, M., Wu, W., Li, L. E., and Mao, Z. M. SoftMoW: Recursive and Reconfigurable Cellular WAN Architecture. In *Proceedings of ACM CoNEXT* (2014), pp. 377–390.

[18] Nokia Networks. LTE network in a box. http://networks.nokia.com/file/40441/lte-network-in-a-box, 2015.

[19] Nokia Siemens Networks. Signaling is growing 50% faster than data traffic. http://networks.nokia.com/system/files%20/document/signaling_whitepaper_online_version_final.pdf, 2012.

[20] Osmani, L., Lindholm, H., Chemmagate, B., Rao, A., Tarkoma, S., Heinonen, J., and Flinck, H. Building blocks for an elastic mobile core. In *Proceedings of the ACM CoNEXT Student Workshop* (2014), pp. 43–45.

[21] Sama, M. R., Contreras, L. M., Kaippallimalil, J., Akiyoshi, I., Qian, H., and Ni, H. Software-defined control of the virtualized mobile packet core. *IEEE Communications Magazine 53*, 2 (Feb 2015), 107–115.

[22] Shenker, S., Casado, M., Koponen, T., McKeown, N., et al. The Future of Networking, and the Past of Protocols. *Open Networking Summit 20* (2011).

[23] Yousaf, F. Z., Lessmann, J., Loureiro, P., and Schmid, S. SoftEPC - Dynamic instantiation of mobile core network entities for efficient resource utilization. In *IEEE International Conference on Communications (ICC)* (2013), pp. 3602–3606.

Appendix C

Bell Labs Prize participation

Our work has been used to participate to the prestigious Bell Labs Prize competition, in which the selected innovative ideas have the opportunity to be implemented together with qualified researchers in Bell Labs. We have submitted a document describing the network-in-a-box idea reported in Section 9.2. Unfortunately, our proposal hasn't passed the selection, but it has been positively acknowledged by Nokia personnel. The submitted document follows.

An *Out of the Box* Software-Defined Cellular Network

Matteo Pozza, Ashwin Rao, and Sasu Tarkoma
University of Helsinki

1 Vision

Cellular networks must be able to evolve to meet the wide range of requirements from various verticals. However, the current LTE networks are largely inflexible [2]. This inflexibility is because the network functions in LTE networks are typically implemented as single function devices which cannot be reprogrammed to perform tasks of other network functions. As a consequence, these single function devices cannot be easily leveraged by network operators to meet the requirements from various verticals.

We believe that network functions should be built using modules that can be dynamically enabled and disabled to allow operators to compose networks according to the demands from their clients. Furthermore, these modularized network functions in turn must be able to interconnect with other network functions and legacy network functions to scale the network.

Our idea is to provide a single box that contains modules which can run the various network functions in LTE networks. Network operators can leverage this single box to bootstrap a cellular network to serve clients who use off-the-shelf LTE enabled devices. Furthermore, operators could compose a network designed to satisfy a specific set of demands by interconnecting these boxes and dynamically enabling and disabling the modules in these boxes. The boxes in turn use Information Centric Networking [3] principles to share the control plane state among themselves. We envision these boxes to be a vital component in future communication networks.

2 Technological Context and Gap Assessment

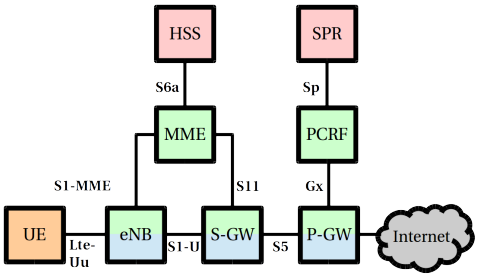


Figure 1: **Current LTE architecture.**

The current LTE architecture (see Figure 1) uses a set of network functions to provide Internet access to user devices. Of the network functions in an LTE network, the Serving Gateways (S-GW), the Packet Data Network Gateway (P-GW), and the eNodeB serve the control plane and the data plane traffic. The control plane decisions are therefore taken at multiple network functions. These network functions share the outcome of the decisions taken by exchanging a wide range of signals to ensure a consistent control plane state. As a consequence, LTE networks are vulnerable to *signaling storms* [4].

The network functions in an LTE network are typically implemented as single function monolithic devices. This static nature of the network functions leads to the ossification of the LTE architecture, which implies the impossibility to change the way the network functions communicate with each other. This also implies the possibility of a road block to address the signaling storm. Furthermore, single function monolithic devices offer limited flexibility to network operators to compose networks tailored for meeting the specific demands from various verticals.

In summary, key network functions in the LTE networks have a convoluted control plane and data plane. Furthermore, network functions are implemented as monolithic devices and this limits the flexibility to compose networks that can evolve. Current LTE networks composed using existing devices are also vulnerable to signaling storms.

3 Preliminary Insights

A starting point to address the above issues is to separate the control plane from the data plane. In particular, our objective is to remove the control logic from the forwarding elements, namely the eNodeBs and S-GW and P-GW, and shift all the control tasks to entities whose aim is to manage these forwarding

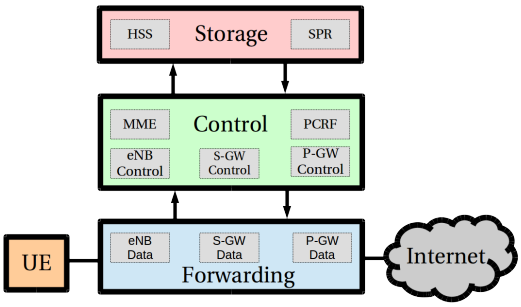


Figure 2: **Abstraction of network functions.** *We envision a Storage Layer, a Control Layer, and a Forwarding Layer.*

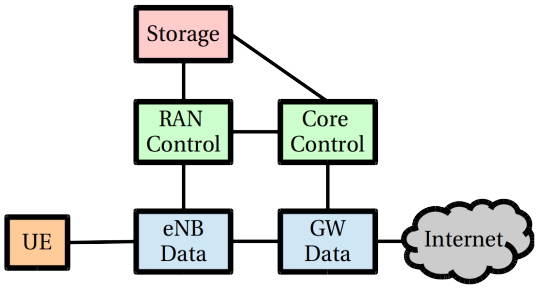
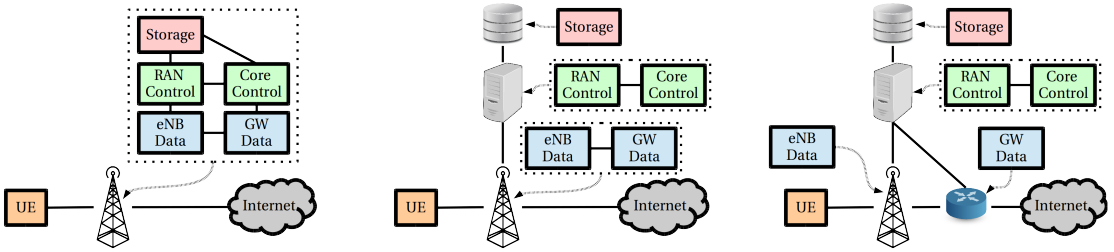


Figure 3: **Refactoring the layers.** *We separate the control plane from the data plane, and the radio access network from the core network.*

elements. This refactoring requires the identification and abstraction of the tasks performed by each network function.

A preliminary analysis of the LTE specifications enabled us to abstract the roles of the network functions into three layers as presented in Figure 2. In particular, we envision a *Storage Layer*, a *Control Layer*, and a *Forwarding layer*. The *Storage Layer* is responsible for persistent data, while the *Control Layer* is responsible for managing and controlling the forwarding elements. These forwarding elements are at the heart of the *Forwarding Layer* which is responsible for forwarding the packets traversing the network. This abstraction enables us to envision forwarding elements whose duties are reduced to implementing the policies mandated by the *Control Layer*. Further refactoring based on these abstractions shows that we can separate the control plane from the data plane, and the radio access network from the core network; we present the modules of this model in Figure 3.

Our refactoring exercise enables us to envision various deployment scenarios, some of which are presented in Figure 4. For example, Figure 4(a) depicts a scenario in which all the modules are present in the same physical device. Similarly, in Figure 4(b) we present a scenario in which we implement the entire control plane on a dedicated server. In some cases it might be useful to keep the radio part of the network separated from its core; such a scenario is depicted in Figure 4(c). In summary, our abstractions can be leveraged to create modules, and the placement of these modules can be determined by the requirements from the network.

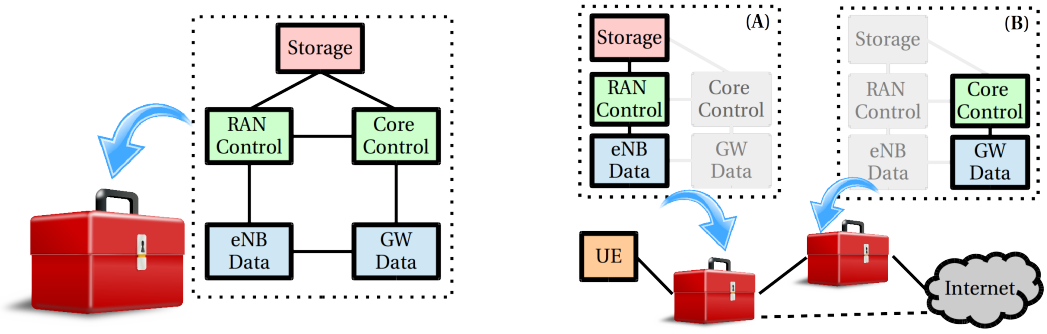


(a) *Modules in a single location.* (b) *Dedicate device for each layer.* (c) *Core in a different location.*

Figure 4: **Deployment scenarios.** *Our abstractions enable us to explore a plethora of scenarios.*

4 Proposed Solution and Implications

We believe that the best way of achieving the maximum level of modularity is by encapsulating these modules in a single physical device, henceforth referred to as a *Box*. Each module in our *Box* (see Figure 5(a)) can be enabled or disabled, and every *Box* can communicate with other *Boxes*. Note that our *Box* does not contain antennas but it will include the option to plug a hardware module for connecting antennas [1]. Furthermore, our *Box* will contain *shim* layers to communicate with legacy devices and network functions from existing LTE networks.



(a) **The *Box* device.** (b) **Example Configuration of *Boxes*.**

Figure 5: *The Box device contains modules that can be dynamically enabled and disabled. Operators can interconnect these Boxes to compose networks tailored to meet the demands from cellular networks. The Boxes will use ICN principles to keep an up to date control plane state to ensure that they can offer services to the devices connected to them when their peers fail.*

The true advantage of having *Boxes* is elasticity. A network operator can customize *Boxes* according the demands from the network. A single *Box* is able to provide a complete ready-to-use cellular network: this is a common use case in natural disasters, where existing cellular network could be seriously compromised. Alternatively, an operator could be interested to dynamically add cellular infrastructure in specific locations to handle crowds in case of sport events and music festivals. Figure 5(b) shows one possible configuration with two *Boxes*: *Box* (A) and *Box* (B). In *Box* (A) we disabled all modules performing the tasks traditionally assigned to the core network, and these modules are enabled in *Box* (B). Furthermore, *Box* (A) can be configured to turn on these modules when *Box* (B) fails.

We propose that these *Boxes* use the principles of Information Centric Networks (ICN) [3] to efficiently disseminate the control plane state to their peers. In particular, each *Box* subscribes to the control plane state information of its clients available at its peers. The *Box* then uses the available information to address failures such as loss of connectivity with a given peer in case of natural disasters. In such a scenario, the clients connected to a given *Box* should be able to communicate with each other even when the *Box* is unable to communicate with its peers.

We envision a two year project to present a version of our *Box*. Our key challenges would be to design and implement the modules of our *Box*, and we plan to work in close collaboration with researchers and engineers from Nokia. Our preliminary insights show that it is indeed possible to refactor the network functions and create the necessary modules. We have shared these insights and have also discussed this proposal with the following experts from Nokia: Hannu Flinck and Gabriel Waller.

Summary. We first abstract the network functions in existing LTE networks. We then leverage our abstractions to refactor the network functions into modules which can be implemented as software modules. These software modules can be placed in a *Box* and can be dynamically enabled and disabled to compose networks that cater to the requirements from network operators. We therefore believe that our solution can be leveraged to create a modular elastic software-defined cellular network that can evolve to serve the changing demands from cellular networks.

References

[1] Endaga - community cellular networks. `https://www.endaga.com/`.

[2] Do we really need new network architecture for 5G? `https://blog.networks.nokia.com/mobile-networks/2015/10/22/really-need-new-network-architecture-5g/`, October 2015.

[3] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking Named Content. In *Proceedings of CoNEXT*. ACM, 2009.

[4] Nokia Siemens Networks. Signaling is growing 50% faster than data traffic. 2012. `http://networks.nokia.com/system/files%20/document/signaling_whitepaper_online_version_final.pdf`.

Appendix D

Chain of Messages

We express in detail the idea of a chain of messages. This is a concept that mainly applies on sequence diagrams, and which shows an important property of the resulting product. We can say that a chain of messages is nothing more than a structured way of building sequences of messages among different entities: what we gain following the structure is that every entity has its sent out messages acknowledged eventually.

For simplicity, we make a distinction between messages that have to be acknowledged (called *commands*) and their acknowledgements. When we make no distinction between commands and acknowledgements, we use the term *messages*. Now consider a generic sequence of messages between two or more elements of the architecture which starts with a command from A to B. The sequence is said to be a chain of messages if these conditions hold:

- the sequence ends with an acknowledgement from B to A;
- A doesn't send out nor receive any message (except for the first command);
- if the subsequence of messages between the first and the last one (excluded) is not empty, then it is composed by one or more chains of messages, each one starting with a command from B to any other element of the architecture (excluding A and all the entities which have a chain of messages still incomplete).

This means that between the reception of the command from A and the acknowledgement sent from B to A, B is allowed to send out a command to another entity C, but it will have to wait for the acknowledgement from C before being allowed to send the acknowledgement back to A. After the interaction with C, B could also send another command to an entity D, but the same condition holds: it has to wait the acknowledgement from D before doing anything else. Figure D.1 depicts an example of chain of messages.

We can notice that, according to the definition, every chain of message has a maximum *nesting level*. The nesting level expresses which is the maximum number of consecutive commands in the subsequence of messages of the chain (i.e. excluding the first command and the last acknowledgement of the chain considered). Informally, it expresses how much deep we are going in our chain before receiving an acknowledgement. We can see that the nesting level of the chain in Figure D.1 is 2, since its subsequence contains 2 chains of nesting level 0 and one chain of nesting level 1.

The main advantage brought by this structure is that every command is acknowledged, possibly in piggyback together with useful data. We prove this property by

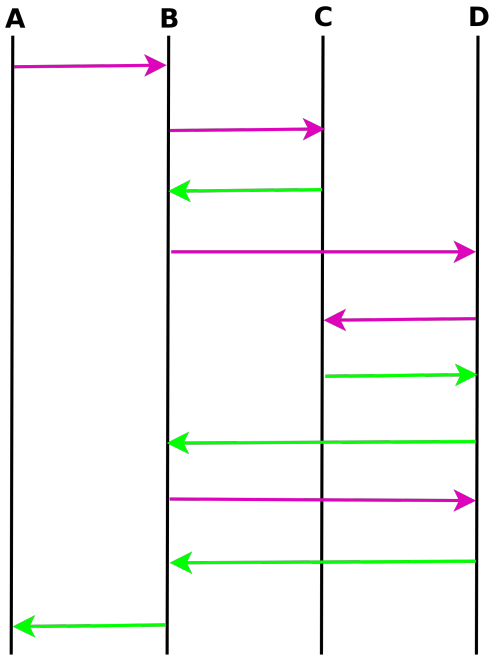


Figure D.1: **An example of chain of messages.** *The commands are in purple, while the acknowledgements are in green.*

induction on the nesting level.

Claim: Chain of messages → Every command is acknowledged

Proof: by induction on the nesting level

Base step: consider a chain of messages of nesting level 0. This means that its subsequence of messages has length 0, otherwise it would be composed by at least a chain of messages of nesting level 0, which would lead it to have nesting level 1. Therefore, the only messages that we have are the first command and the last acknowledgement, which actually acknowledges the fist command. Therefore, we cannot have unacknowledged commands.

Inductive step: suppose that the property holds for chain of messages till nesting level k, and we consider a chain of messages of nesting level $k+1$. This implies that its subsequence of messages is composed by one or more chains of messages whose nesting level is less or equal to k. We need to prove that the sequential combination of chains of messages that have the property of having every command acknowledged produces a sequence of messages in which every command is acknowledged.

Corollary: The sequential combination of chain of messages in which every command is acknowledged produces a sequence of messages in which every command is acknowledged.

Proof: A sequential combination simply means that we are putting the chains of messages one after the other on the time axis (the vertical one), without mixing their messages, but just putting them sequentially. Since this operation brings no new messages to the whole sequence, and every command was acknowledged in each of the chains of messages considered, the resulting sequence of messages has the property of

having every command acknowledged as well.

Now that we have proven the needed property, we can combine it with the inductive hypothesis, which says that a chain of messages of nesting level less or equal to k has the property of having every command acknowledged. This implies that the subsequence of messages of the chain of messages of nesting level $k + 1$ has actually the property of having every command acknowledged. We are adding to the subsequence of messages a command on top of it, which is acknowledged by the acknowledgement we are adding at the bottom of the subsequence. By the definition of chain of message, we know that there are no messages going to the entity issuing the first command, nor messages departing from it, therefore there are no messages related to this entity except for the first and the last one of the chain of messages. Therefore, we are sure that the resulting chain of messages has no unacknowledged commands in it.

This property is particularly important since it says that if we build a chain of message, instead of an unstructured sequence of messages, we can be sure that all the commands are acknowledged, and therefore we are computing the worst-case of messages needed in our communication in order to make it reliable.

Now consider the sequences of messages of the procedures belonging to our final model. First, we exclude from our analysis the messages exchanged between eNB and UE. Second, we exclude the messages that consist in the "notification" to the RAN Control, *i.e.* all the messages from the beginning till the first one that arrives to the RAN Control included. The remaining sequence of messages is actually a sequential combination of chains of messages, which we have proven to have the property of having every command acknowledged.

Bibliography

[1] Crawdad - a community resource for archiving wireless data at dartmouth. http://www.crawdad.org/.

[2] Endaga - community cellular networks. https://www.endaga.com/.

[3] Loon for all - baloon powered internet for everyone. https://www.google.com/loon/.

[4] Lte-epc network simulator (lena). http://networks.cttc.es/mobile-networks/software-tools/lena/.

[5] Lte-sim - telematics - politecnico di bari. http://telematics.poliba.it/index.php/en/lte-sim.

[6] nwepc - epc sae gateway. https://sourceforge.net/projects/nwepc/.

[7] Open5gcore – the next mobile core network testbed platform. http://www.open5gcore.org/.

[8] Openairinterface - 5g software alliance for democratising wireless innovation. http://www.openairinterface.org/.

[9] Openepc – the openepc project. http://www.openepc.com/.

[10] Openlte - open source implementation of the 3gpp lte specifications. http://openlte.sourceforge.net/.

[11] Opensdncore – research & testbed for the carrier-grade nfv/sdn environment. http://www.opensdncore.org/.

[12] Phantomnet testbed. https://www.phantomnet.org/.

[13] Simulte - lte user plane simulator for omnet++ and inet. http://simulte.com/.

[14] Vienna lte-a simulators. https://www.nt.tuwien.ac.at/research/mobile-communications/vienna-lte-a-simulators/.

[15] 3GPP. Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C). TS 29.274, 3rd Generation Partnership Project (3GPP), December 2015.

[16] 3GPP. General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access. TS 23.401, 3rd Generation Partnership Project (3GPP), December 2015.

[17] 3GPP. Home Subscriber Server (HSS) diameter interfaces for interworking with packet data networks and applications. TS 29.336, 3rd Generation Partnership Project (3GPP), December 2015.

[18] 3GPP. Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol. TS 29.272, 3rd Generation Partnership Project (3GPP), December 2015.

[19] 3GPP. Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS). TS 24.301, 3rd Generation Partnership Project (3GPP), December 2015.

[20] 3GPP. Policy and charging control architecture. TS 23.203, 3rd Generation Partnership Project (3GPP), December 2015.

[21] 3GPP. S1 Application Protocol (S1AP). TS 36.413, 3rd Generation Partnership Project (3GPP), December 2015.

[22] 3GPP. Security architecture. TS 33.401, 3rd Generation Partnership Project (3GPP), December 2015.

[23] 3GPP. X2 Application Protocol (X2AP). TS 36.423, 3rd Generation Partnership Project (3GPP), December 2015.

[24] 3GPP. Physical channels and modulation. TS 36.211, 3rd Generation Partnership Project (3GPP), January 2016.

[25] 3GPP. Radio Resource Control (RRC); Protocol specification. TS 36.331, 3rd Generation Partnership Project (3GPP), January 2016.

[26] 3rd Generation Partnership Project. Lte overview. http://www.3gpp.org/technologies/keywords-acronyms/98-lte.

[27] J. B. Abdo, I. Sarji, I. H. Elhajj, A. Chehab, and A. Kayssi. Application-aware fast dormancy in lte. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 194–201, May 2014.

[28] Alcatel-Lucent. 7750 Service Router - Mobile Gateway. https://www.alcatel-lucent.com/products/7750-service-router-mobile-gateway.

[29] Alcatel-Lucent. Managing the signaling traffic in packet core. 2012. http://resources.alcatel-lucent.com/asset/155160.

[30] X. An, F. Pianese, I. Widjaja, and U. G. Acer. dmme: Virtualizing lte mobility management. In *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pages 528–536, Oct 2011.

[31] Nicola Baldo, Marco Miozzo, Manuel Requena-Esteso, and Jaume Nin-Guerrero. An open source product-oriented lte network simulator based on ns-3. In *Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '11, pages 293–298, New York, NY, USA, 2011. ACM.

[32] Nicola Baldo, Manuel Requena-Esteso, Marco Miozzo, and Raymond Kwan. An open source model for the simulation of lte handover scenarios and algorithms in ns-3. In *Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems*, MSWiM '13, pages 289–298, New York, NY, USA, 2013. ACM.

[33] Arijit Banerjee, Rajesh Mahindra, Karthik Sundaresan, Sneha Kasera, and Jacobus Van der Merwe and Sampath Rangarajan. Scaling the lte control-plane for future mobile access. In *Proceedings of the Eleventh ACM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*, December 2015.

[34] Arsany Basta, Wolfgang Kellerer, Marco Hoffmann, Hans Jochen Morper, and Klaus Hoffmann. Applying nfv and sdn to lte mobile core gateways, the functions placement problem. In *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges*, AllThingsCellular '14, pages 33–38, New York, NY, USA, 2014. ACM.

[35] Nec Corporation. Nec virtualized evolved packet core – vepc. 2015. http://www.nec.com/en/global/solutions/tcs/pdf/vEPC_WP.pdf.

[36] Radisys Corporation. Protocol signaling procedures in lte. 2011. http://go.radisys.com/rs/radisys/images/paper-lte-protocol-signaling.pdf.

[37] J. Costa-Requena, R. Kantola, J. Llorente, V. Ferrer, J. Manner, A. Y. Ding, Y. Liu, and S. Tarkoma. Software defined 5g mobile backhaul. In *5G for Ubiquitous Connectivity (5GU), 2014 1st International Conference on*, pages 258–263, Nov 2014.

[38] Diametriq. Measuring the explosion of lte signaling traffic - a diameter traffic model. 2012. http://diametriq.com/wp-content/uploads/downloads/2012/09/A-Diameter-Traffic-Model.pdf.

[39] Diametriq. A storm is brewing – an lte signaling storm. 2012. http://www.diametriq.com/wp-content/uploads/downloads/2013/01/A-Storm-is-Brewing.pdf.

[40] W. Diego, I. Hamchaoui, and X. Lagrange. Cost factor analysis of qos in lte/epc mobile networks. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 614–619, Jan 2016.

[41] NTT Docomo. Security technology for sae/lte. 2009. https://www.nttdocomo.co.jp/english/binary/pdf/corporate/technology/rd/technical_journal/bn/vol11_3/vol11_3_027en.pdf.

[42] Netmanias Technical Document. Eleven emm cases in an emm scenario. October 2013. http://www.netmanias.com/en/post/techdocs/6002/emm-lte/eleven-emm-cases-in-an-emm-scenario.

[43] Netmanias Technical Document. Emm procedure 1. initial attach - part 1. December 2013. http://www.netmanias.com/en/?m=view&id=techdocs_mbl&page=2&no=6098.

[44] Netmanias Technical Document. Lte emm and ecm states. September 2013. http://www.netmanias.com/en/post/techdocs/5909/ecm-emm-lte-mobility/lte-emm-and-ecm-states.

[45] Netmanias Technical Document. Lte identification i : Ue and me identifiers. August 2013. www.netmanias.com/en/post/techdocs/5905/identification-lte/lte-identification-i-ue-and-me-identifiers.

[46] Netmanias Technical Document. Lte identification ii : Ne and location identifiers. August 2013. http://www.netmanias.com/en/post/techdocs/5906/identification-identifier-lte/lte-identification-ii-ne-and-location-identifiers.

[47] Netmanias Technical Document. Lte identification iii: Eps session/bearer identifiers. August 2013. http://www.netmanias.com/en/post/techdocs/5907/identification-identifier-lte/lte-identification-iii-eps-session-bearer-identifiers.

[48] Netmanias Technical Document. Lte network architecture: Basic. July 2013. http://www.netmanias.com/en/post/techdocs/5904/architecture-lte/lte-network-architecture-basic.

[49] Netmanias Technical Document. Lte qos: Sdf and eps bearer qos. September 2013. http://www.netmanias.com/en/post/techdocs/5908/eps-lte-qos-sdf/lte-qos-sdf-and-eps-bearer-qos.

[50] Netmanias Technical Document. Lte security i: Lte security concept and lte authentication. July 2013. http://www.netmanias.com/en/post/techdocs/5902/lte-security/lte-security-i-concept-and-authentication.

[51] Netmanias Technical Document. Lte security ii: Nas and as security. August 2013. http://www.netmanias.com/en/post/techdocs/5903/lte-security/lte-security-ii-nas-and-as-security.

[52] Netmanias Technical Document. Lte: Tracking area (ta) and tracking area update (tau). August 2013. http://www.netmanias.com/en/post/blog/5930/lte-tracking-area/lte-tracking-area-ta-and-tracking-area-update-tau.

[53] Netmanias Technical Document. Emm procedure 1. initial attach - part 2. January 2014. http://www.netmanias.com/en/?m=view&id=techdocs_mbl&page=2&no=6102.

[54] Netmanias Technical Document. Emm procedure 3. s1 release. January 2014. http://www.netmanias.com/en/?m=view&id=techdocs_mbl&page=2&no=6110.

[55] Netmanias Technical Document. Emm procedure 4. service request. February 2014. http://www.netmanias.com/en/?m=view&id=techdocs_mbl&page=2&no=6134.

[56] Netmanias Technical Document. Emm procedure 6. handover without tau - part 1. overview of lte handover. March 2014. http://www.netmanias.com/en/?m=view&id=techdocs_mbl&page=2&no=6224.

[57] Netmanias Technical Document. Emm procedure 6. handover without tau - part 2. x2 handover. March 2014. http://www.netmanias.com/en/?m=view&id=techdocs_mbl&page=2&no=6257.

[58] Netmanias Technical Document. Emm procedure 6. handover without tau - part 3. s1 handover. April 2014. http://www.netmanias.com/en/?m=view&id=techdocs_mbl&page=2&no=6286.

[59] Ericsson. Everything you ever wanted to know about lte. 2011. http://www.comlab.uniroma3.it/ens/LTE2.pdf.

[60] Ericsson. Cloud ran - the benefits of virtualization, centralization and coordination. 2015. https://www.ericsson.com/res/docs/whitepapers/wp-cloud-ran.pdf.

[61] Ericsson. Ericsson mobility report. 2015. https://www.ericsson.com/res/docs/2015/ericsson-mobility-report-june-2015.pdf.

[62] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The road to sdn. *Queue*, 11(12):20:20–20:40, December 2013.

[63] Nate Foster, Rob Harrison, Michael J. Freedman, Christopher Monsanto, Jennifer Rexford, Alec Story, and David Walker. Frenetic: A network programming language. *SIGPLAN Not.*, 46(9):279–291, September 2011.

[64] NMC Consulting Group. Lte identifiers. 2011. http://www.nmcgroups.com/files/download/NMC.LTE%20Identifiers.v1.0.pdf.

[65] Aditya Gudipati, Daniel Perry, Li Erran Li, and Sachin Katti. Softran: Software defined radio access network. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 25–30, New York, NY, USA, 2013. ACM.

[66] Junxian Huang, Feng Qian, Alexandre Gerber, Z. Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 225–238, New York, NY, USA, 2012. ACM.

[67] Huawei. Smartphone solutions white paper. 2012. http://www.huawei.com/mediafiles/CBG/PDF/Files/hw_193034.pdf.

[68] J.C. Ikuno, M. Wrulich, and M. Rupp. System level simulation of lte networks. In *Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st*, pages 1–5, May 2010.

[69] European Telecommunications Standards Institute. Network functions virtualisation (nfv); use cases. 2013. http://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.01_60/gs_nfv001v010101p.pdf.

[70] Xin Jin, Li Erran Li, Laurent Vanbever, and Jennifer Rexford. Softcell: Scalable and flexible cellular core network architecture. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, pages 163–174, New York, NY, USA, 2013. ACM.

[71] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 654–663, New York, NY, USA, 1997. ACM.

[72] J. Kempf, B. Johansson, S. Pettersson, H. Lüning, and T. Nilsson. Moving the mobile evolved packet core to the cloud. In *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 784–791, Oct 2012.

[73] L. E. Li, Z. M. Mao, and J. Rexford. Toward software-defined cellular networks. In *2012 European Workshop on Software Defined Networking*, pages 7–12, Oct 2012.

[74] Heikki Lindholm, Lirim Osmani, Hannu Flinck, Sasu Tarkoma, and Ashwin Rao. State space analysis to refactor the mobile core. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, AllThingsCellular '15, pages 31–36, New York, NY, USA, 2015. ACM.

[75] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[76] Esa Markus Metsälä and Juha Salmelin. *LTE Backhaul - Planning and Optimization*. Wiley, 2015.

[77] China Mobile. C-ran: the road towards green radio access network. 2012. http://labs.chinamobile.com/cran/wp-content/uploads/CRAN_white_paper_v2_5_EN.pdf.

[78] Mehrdad Moradi, Wenfei Wu, Li Erran Li, and Zhuoqing Morley Mao. Softmow: Recursive and reconfigurable cellular wan architecture. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, pages 377–390, New York, NY, USA, 2014. ACM.

[79] D. Naboulsi, M. Fiore, S. Ribot, and R. Stanica. Large-scale mobile traffic analysis: A survey. *IEEE Communications Surveys Tutorials*, 18(1):124–161, Firstquarter 2016.

[80] Nokia Networks. What is going on in mobile broadband networks? 2014. http://networks.nokia.com/system/files/document/nokia_smartphone_traffic_white_paper.pdf.

[81] Nokia Networks. Lte network in a box. 2015. http://networks.nokia.com/sites/default/files/document/nokia_network_in_a_box_white_paper_0.pdf.

[82] Nokia Networks. Lte network in a box: For public safety mobile broadband. 2015. http://networks.nokia.com/sites/default/files/document/nib_slolutions_public_safety_whitepaper.pdf.

[83] Nokia Siemens Networks. More battery life with lte connected state drx. http://networks.nokia.com/system/files/document/more_battery_life_with_lte_connected_drx_0.pdf.

[84] Nokia Siemens Networks. Signaling is growing 50% faster than data traffic. 2012. http://networks.nokia.com/system/files%20/document/signaling_whitepaper_online_version_final.pdf.

[85] Navid Nikaein, Eryk Schiller, Romain Favraud, Kostas Katsalis, Donatos Stavropoulos, Islam Alyafawi, Zhongliang Zhao, Torsten Braun, and Thanasis Korakis. Network store: Exploring slicing in future 5g networks. In *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*, MobiArch '15, pages 8–13, New York, NY, USA, 2015. ACM.

[86] Navid Nikaein, Eryk Schiller, Romain Favraud, Raymond Knopp, Islam Alyafawi, and Torsten Braun. *Towards a cloud-native radio access network*. Book Chapter of "Studies in Big Data", Springer, 2016, 02 2016.

[87] Oracle. The new diameter network: Managing the signaling storm. 2013. http://www.oracle.com/us/industries/communications/managing-signaling-storm-wp-2101039.pdf.

[88] G. Piro, L.A. Grieco, G. Boggia, F. Capozzi, and P. Camarda. Simulating lte cellular systems: An open-source framework. *Vehicular Technology, IEEE Transactions on*, 60(2):498–513, Feb 2011.

[89] Matteo Pozza. Current lte architecture - active to idle transition - slides. https://drive.google.com/open?id=0B49I92-bJ59KdEpHaFVLRmgzUTA.

[90] Matteo Pozza. Current lte architecture - idle to active transition (network triggered) - slides. https://drive.google.com/open?id=0B49I92-bJ59KLVczNXVfVU8xSzA.

[91] Matteo Pozza. Current lte architecture - idle to active transition (ue triggered) - slides. https://drive.google.com/open?id=0B49I92-bJ59KOGo2Ym1fT3VTZFE.

[92] Matteo Pozza. Current lte architecture - initial attach - slides. https://drive.google.com/open?id=0B49I92-bJ59Ka1FsQndKSDF5Ymc.

[93] Matteo Pozza. Current lte architecture - s1 handover - slides. https://drive.google.com/open?id=0B49I92-bJ59KWUt4RWxzbkVjRjA.

[94] Matteo Pozza. Current lte architecture - slides folder. https://drive.google.com/open?id=0B49I92-bJ59Kd3p0TWtXazRoQnc.

[95] Matteo Pozza. Current lte architecture - x2 handover - slides. https://drive.google.com/open?id=0B49I92-bJ59KRWtRNklvNW5vbU0.

[96] Matteo Pozza. Refactoring step 1 - active to idle transition - slides. https://drive.google.com/open?id=0B49I92-bJ59KMDQyZXQtNDNENlU.

[97] Matteo Pozza. Refactoring step 1 - idle to active transition (network triggered) - slides. https://drive.google.com/open?id=0B49I92-bJ59Kd3dqaVgwdnNOQjg.

[98] Matteo Pozza. Refactoring step 1 - idle to active transition (ue triggered) - slides. https://drive.google.com/open?id=0B49I92-bJ59KWHlPTTBWRmpkcXM.

[99] Matteo Pozza. Refactoring step 1 - initial attach - slides. https://drive.google.com/open?id=0B49I92-bJ59KdTg3SWRrelprWXM.

[100] Matteo Pozza. Refactoring step 1 - s1 handover - slides. https://drive.google.com/open?id=0B49I92-bJ59KV1Z5ajE4a3VBSFE.

[101] Matteo Pozza. Refactoring step 1 - slides folder. https://drive.google.com/open?id=0B49I92-bJ59KRG9ObFNveXp3S2s.

[102] Matteo Pozza. Refactoring step 1 - x2 handover - slides. https://drive.google.com/open?id=0B49I92-bJ59KQl9UWHFNTXphME0.

[103] Matteo Pozza. Refactoring step 2 - active to idle transition - slides. https://drive.google.com/open?id=0B49I92-bJ59KVGg0eUtVOEpqRnM.

[104] Matteo Pozza. Refactoring step 2 - idle to active transition (network triggered) - slides. https://drive.google.com/open?id=0B49I92-bJ59KM2xmcUh1MFN6Tnc.

[105] Matteo Pozza. Refactoring step 2 - idle to active transition (ue triggered) - slides. https://drive.google.com/open?id=0B49I92-bJ59KelhYN0JCV3hQRmc.

[106] Matteo Pozza. Refactoring step 2 - initial attach - slides. https://drive.google.com/open?id=0B49I92-bJ59KVUduR2Eyd3ZrNkU.

[107] Matteo Pozza. Refactoring step 2 - s1 handover - slides. https://drive.google.com/open?id=0B49I92-bJ59Kb01nTHoyT2xPeFk.

[108] Matteo Pozza. Refactoring step 2 - slides folder. https://drive.google.com/open?id=0B49I92-bJ59KRmtwdFFSNE5hV1U.

[109] Matteo Pozza. Refactoring step 2 - x2 handover - slides. https://drive.google.com/open?id=0B49I92-bJ59KclVGcFgtVEtxTnM.

[110] Matteo Pozza. Refactoring step 3 - active to idle transition - slides. https://drive.google.com/open?id=0B49I92-bJ59Kbk9MR3k4OGpsS3M.

[111] Matteo Pozza. Refactoring step 3 - handover - slides. https://drive.google.com/open?id=0B49I92-bJ59KLXVReE83VXBka3M.

[112] Matteo Pozza. Refactoring step 3 - idle to active transition (network triggered) - slides. https://drive.google.com/open?id=0B49I92-bJ59KRTdEWjZ3Tzk2Qms.

[113] Matteo Pozza. Refactoring step 3 - idle to active transition (ue triggered) - slides. https://drive.google.com/open?id=0B49I92-bJ59Kb01GZ01aT1RNMWM.

[114] Matteo Pozza. Refactoring step 3 - initial attach - slides. https://drive.google.com/open?id=0B49I92-bJ59KUHFGZEpkVk9kTGM.

[115] Matteo Pozza. Refactoring step 3 - slides folder. https://drive.google.com/open?id=0B49I92-bJ59KejVEQ3lhUS1GRWs.

[116] Zafar Ayyub Qazi, Phani Krishna, Vyas Sekar, Vijay Gopalakrishnan, Kaustubh Joshi, and Samir R Das. Klein: A minimally disruptive design for an elastic cellular core.

[117] M. R. Sama, L. M. Contreras, J. Kaippallimalil, I. Akiyoshi, H. Qian, and H. Ni. Software-defined control of the virtualized mobile packet core. *IEEE Communications Magazine*, 53(2):107–115, Feb 2015.

[118] Florin Sandu, Szilard Cserey, and Eugen Mile-Ciobanu. Simulation of lte signaling. *Advances in Electrical and Computer Engineering*, 10(2):108–114, 2010.

[119] Freescale Semiconductor. Long term evolution protocol overview. 2008. https://www.nxp.com/files/wireless_comm/doc/white_paper/LTEPTCLOVWWP.pdf.

[120] Spirent. Signaling storms push diameter networks to the limit. 2015. http://www.spirent.com/-/media/eBooks/Signaling_storms_push_Diameter_networks_to_the_limit_eBook.pdf.

[121] Tonse Telecom. The lte data storm in the core of your network! 2013. http://www.tonsetelecom.com/sites/default/files/The%20LTE%20Data%20Storm%20in%20the%20Core%20of%20Your%20Network_Jan_8_2013.pdf.

[122] Nathanael Thompson, Petros Zerfos, Robert Sombrutzki, Jens-Peter Redlich, and Haiyun Luo. 100networks. In *Proceedings of the 9th Workshop on Mobile Computing Systems and Applications*, HotMobile '08, pages 27–32, New York, NY, USA, 2008. ACM.

[123] Antonio Virdis, Giovanni Stea, and Giovanni Nardini. Simulte-a modular system-level simulator for lte/lte-a networks based on omnet++. In *2014 International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)*, pages 59–70. IEEE, 2014.

[124] I. Widjaja, P. Bosch, and H. La Roche. Comparison of mme signaling loads for long-term-evolution architectures. In *Vehicular Technology Conference Fall (VTC 2009-Fall), 2009 IEEE 70th*, pages 1–5, Sept 2009.

[125] F. Z. Yousaf, J. Lessmann, P. Loureiro, and S. Schmid. Softepc - dynamic instantiation of mobile core network entities for efficient resource utilization. In *2013 IEEE International Conference on Communications (ICC)*, pages 3602–3606, June 2013.

Acronyms

3GPP 3rd Generation Partnership Project.

AMBR Aggregated Maximum Bit Rate.

APN Access Point Name.

ARP Allocation and Retention Priority.

AS Access Stratum.

ASME Access Security Management Entity.

AtI Active to Idle.

BH Busy Hour.

BS Base Station.

C-RNTI Cell Radio Network Temporary Identifier.

CAPEX CAPital EXpenditure.

CDF Cumulative Distribution Function.

CRC Cyclic Redundancy Check.

DL Download.

DPI Deep Packet Inspection.

DRB Data Radio Bearer.

DRX Discontinuous Reception.

E-RAB E-UTRAN Radio Access Bearer.

E-UTRAN Evolved UMTS Terrestrial Radio Access Network.

ECGI E-UTRAN Cell Global Identifier.

ECM EPS Connection Management.

eNB E-UTRAN Node B.

EPC Evolved Packet Core.

EPS Evolved Packet System.

GBR Guaranteed Bit Rate.

GPRS General Packet Radio Service.

GTP GPRS Tunneling Protocol.

GUTI Globally Unique Temporary ID.

GW Gateway.

HD High-Definition.

HSPA High-Speed Packet Access.

HSS Home Subscriber Server.

IA Initial Attach.

ICN Information-Centric Networking.

IMSI International Mobile Subscriber Identity.

ISP Internet Service Provider.

ItA Idle to Active.

JVM Java Virtual Machine.

KSI Key Set Identifier.

LTE Long Term Evolution.

MAC Medium Access Control.

MME Mobile Management Entity.

NAS Non-Access Stratum.

NAT Network Address Translation.

NFV Network Function Virtualization.

NOS Network Operating System.

OPEX OPerating EXpenditure.

P-GW PDN Gateway.

PCC Policy Control and Charging.

PCRF Policy and Charging Rules Function.

PDN Packet Data Network.

PHY Physical.

PLMN Public Land Mobile Network.

QCI QoS Class Identifier.

QoS Quality of Service.

RAN Radio Access Network.

RES Response.

RRC Radio Resource Control.

S-GW Serving Gateway.

S1AP S1 Application Protocol.

S1H S1 Handover.

SAU Simultaneous Attached Users.

SDF Service Data Flow.

SDN Software-Defined Networking.

SeNB Source eNB.

SON Self-Organizing Network.

SPR Subscription Profile Repository.

TA Tracking Area.

TAI Tracking Area Identifier.

TAU Tracking Area Update.

TCP Transmission Control Protocol.

TEID Tunnel Endpoint IDentifier.

TeNB Target eNB.

TFT Traffic Flow Template.

UE User Equipment.

UL Upload.

UMTS Universal Mobile Telecommunication System.

WAN Wide Area Network.

WLAN Wireless Local Area Network.

X2H X2 Handover.

XRES Expected Response.