A study of data consistency constraints in 5G, applied to limiting resource usage in network slices

Jonathan Sid-Otmane

Thesis submitted to obtain the degree of doctor of philosophy from

Sorbonne Université

École doctorale ÉDITE de Paris (ED130)
Informatique, Télécommunication et Électronique

Laboratoire d'Informatique de Paris 6 (LIP6)
Orange Labs

A study of data consistency constraints in 5G, applied to limiting resource usage in network slices

---

Jonathan Sid-Otmane

Defended on the 13th of December 2021, before a jury composed of:

Jury members:

Brigitte Kervella, Associate Professor, Sorbonne Université	*Examiner*
Julien Maisonneuve, Standardisation Manager, Nokia	*Examiner*
Sébastien Monnet, Full Professor, Université Savoie Mont Blanc	*Examiner*
Pierre Sens, Full Professor, Sorbonne Université	*Examiner*
Samia Bouzefrane, Full Professor, CNAM	*Reviewer*
Cédric Tedeschi, Associate Professor, University of Rennes I	*Reviewer*
Sofiane Imadali, Research Scientist, Orange Labs	*Advisor*
Frédéric Martelli, Service Architect, Orange Labs	*Advisor*
Marc Shapiro, Distinguished Research Scholar, Sorbonne Université & Inria	*Advisor*

Remerciements

Cette thèse n'aurait pas été possible sans le soutien de nombreuses personnes. J'ai tenté d'en dresser ici la liste exhaustive,[1] mais je ne doute pas d'en avoir oublié. Je vais donc commencer par remercier tous ceux qui ne figurent pas sur cette page, je m'excuse d'avance d'avoir oublié d'y inscrire votre nom. C'était un instant d'étourderie et je ne manquerai pas de vous remercier de vive voix lors de notre prochaine rencontre.

Pour le temps qu'il m'a consacré durant toutes ces années, la qualité de ses conseils à chaque échange, et sa grande disponibilité durant cette période de rédaction, je remercie mon directeur de thèse Marc Shapiro.

Je tiens également à remercier mes deux encadrants, Sofiane Imadali et Frédéric Martelli, pour leur accompagnement durant ce projet. Je reste étonné par leur implication et l'énergie qu'ils ont déployée pour me faire terminer cette thèse.

Je remercie les membres du jury pour l'intérêt et le temps consacrés à mes travaux. Particulièrement Samia Bouzefrane et Cédric Tedeschi qui ont accepté le rôle de rapporteur. Je remercie Brigitte Kervella et Sébastien Monnet, qui ont de surcroît composé mon comité de suivi de thèse, et ont vu évoluer celle-ci.

J'ai passé ces années à la frontière entre deux mondes, et ce sont les membres d'Iron chez Orange qui m'ont accompagné durant ma découverte des télécoms et du monde de l'entreprise: Je remercie Aranzazu, Ayoub, Eftychia, Grégory, Juan, Malika, Marion, et Xavier.

Je n'oublie pas Louiza Yala et Sihem Cherrared (ma co-encadrée) merci pour nos moments passés ensemble, tous vos précieux conseils dictés par la sagesse que confère la séniorité d'un doctorat déjà obtenu ou d'une année de plus passée en thèse.

Pour le LIP6, je tiens d'abord à remercier les permanents. Enseignants devenus collègues, leur accueil au LIP6 lors d'un stage puis d'une thèse mérite déjà ma reconnaissance, mais avant tout ils ont su me donner l'envie de faire de l'informatique et de la recherche. L'enseignement a aussi été une belle découverte, je remercie ceux

[1] Sur les conseils d'un sage docteur, l'ensemble des listes de cette section ont été classées alphabétiquement.

qui m'ont permis de la faire et aux côtés desquels j'ai enseigné: Cassandre, Cédric, Jonathan, Julien, Luciana, Pierre, Swann, et Yann.

Pour les jeunes et futurs docteurs du labo, tout ce temps passé ensemble ne se résume pas à des pauses café parfois trop longues. Mes rencontres avec d'autres thésards, qui n'ont pas eu la chance de vous avoir, montrent que la thèse est une tâche difficile parfois rendue impossible par l'isolement. Votre implication pour tous les autres membres lorsqu'ils rencontrent un écueil n'est pas seulement rare, elle est surtout indispensable: merci Antoine, Arnaud, Aymeric, Benoit, Célia, Dimitrios, Élise, Étienne, Florent, Francis, Gabriel, Guillaume, Hakan, Ilyas, Laurent, Lucas, Marjorie, Maxime, Sreeja, et Vincent. Une pensée particulière pour ceux qui ont un jour partagé avec moi le Bureau 215: Alejandro, Ludovic (même si ce fut très court), Saalik, et Sara.

La vie, c'est avant tout des rencontres, et en accumulant les années à la fac, j'ai aussi eu le bonheur d'accumuler les rencontres: Alexandra, Aymeric, Claire-Marine, Jean-Christophe, Julie, Julien, Kévin, Massy, Medhi, Michael, Nouredine, Olivia, Oskar, Senda, Victor et Yassine. Certaines de ces rencontres on fait la même erreur que moi: Ludovic, Léo, Manon, Romain, Saalik, Sofiane, et Stratis. On s'en sera bientôt tous sortis, courage aux deux qui restent!

Il y a aussi ces recontres qui remontent à si loin qu'on ne se souvient plus de quand ça a commencé. Je remercie donc ceux que j'ai rencontrés bien avant : Alexis, Alice, Anne, Ouçama, Aïness, Emna, et Medhi.

Je ne sais pas ce que je ferais sans vous, et j'aurais aimé trouver une phrase drôle et originale pour vous le dire.

La liste de ce que je dois à ma famille est longue, et cette section est trop courte pour les remercier correctement. Dans un soucis de brièveté, je ne nommerais que mes parents, ma soeur Hanna, et ma tante Zohra.

Pour tous les autres, je vous invite à vous référer au premier paragraphe.

Abstract

The 5G network specification requires ACID properties (Atomicity, Consistency, Isolation, Durability) for its distributed database. According to the CAP theorem, these properties are incompatible with high availability for a geo-distributed system. Since availability is an obligation for mobile networks, this contradiction challenges the specification.

Our thesis focus on a study of the usage of data in the network that allow us to extract weaker consistency properties that are still sufficient to maintain the correctness of the data.

In the case a Network Slicing, an innovation of the 5G network, we propose to study further the enforcement of one of these properties, through the example of the enforcement of a global limit on the resource usage of a slice.

Slices are deployed over a potentially large are, and so is their data, which make the enforcement of a limit more challenging. We propose several algorithms placed at different points of the consistency spectrum and study their performance in a simulation of the 5G network.

Résumé

Si les spécifications de la 5G sont toujours en développement, il est clair que le système de support de la 5G constituera un véritable système géo-distribué, avec donc des problématiques de cohérence. Le sujet de cette thèse est d'étudier la spécification de ce point de vue, et d'analyser des propriétés de cohérence nécessaires.

Notre première contribution se base sur une analyse approfondie des procédures du réseau mobile 5G, connexion initiale, établissement de session, migration du terminal. Ces procédures accèdent aux données spécifiques à l'utilisateur, appelées UEC (User Equipment Context). D'après la spécification, l'UEC est stockée dans une base de données ayant des propriétés ACID (Atomicity, Consistency, Isolation, Durability). Or, ACID n'est pas compatible avec un déploiement géo-distribué de haute disponibilité. Les réseaux mobiles sont sujets à des obligations de disponibilité, cette contradiction remet en question la spécification. Dans un article, publié à ICIN2020, nous établission les propriétés de cohérence suffisantes, plus faibles qu'ACID, qui assurent que les données resteront correctes [27].

Notre seconde contribution étudie les données distribuées liées aux tranches (Slices), une innovation de la 5G. Les tranches partitionnent le réseau physique en réseaux virtuels, de manière dynamique. Une tranche offre des garanties spécifiques (par exemple de latence plus fortes, ou la possibilité de s'adresser à une flotte d'appareils). Une tranche est un objet virtuel, qui doit faire respecter des contraintes associées de façon distribuée. Nous examinons en particulier le cas d'une tranche qui doit borner globalement la consommation d'une ressource, grâce à un objet réparti, le limiteur. Celui-ci s'appuie sur des données distribuées, posant donc des problématiques de cohérence.

Puisque la 5G n'est pas encore déployée, et vu qu'il est difficile de prévoir les futurs usages du réseau, nous avons créé un simulateur de tranches. Afin d'étudier les bornes d'utilisation des ressources réseau. Nous évaluons plusieurs algorithmes différents de limiteur et proposons une méthode de mesure de leur performance selon les besoins remplis par chaque tranche.

Contents

1 Introduction

The 5G specification describes a Software Defined Networking (SDN) based, virtualized network of Virtual Network Functions (VNFs) distributed over large geographical areas. The specification as it stands makes consistency requirements that are not compatible with a resilient mobile network. Our thesis suggests instead to relax the consistency semantics of the network, except where necessary to preserve correctness.

5G is the newest installment of an ever progressing improvement that has taken place since the first days of mobile connectivity. The evolution of mobile networks takes two parallel tracks. On the one hand, antennas evolve and new radio protocols are brought into production, this allows mobile terminals connections to be easier, more resilient and provides more bandwidth for content. On the other hand, less visible to end users, but no less important, the evolution of the core network *i.e.* the Telecom Operators' infrastructure that support mobile users and their new needs, to support the augmented capabilities of both antennas and terminals.

The design of each new generation of mobile networks is the result of a global effort by industrial actors to fulfill needs that were not foreseen during the design of the previous generation. Global System for Mobile Communications (GSM) networks were focused on allowing voice connectivity. General Packet Radio Service (GPRS) was the first network that introduced packet switching in mobile networks, where previously Mobile Network Operators (MNOs) had relied on *circuit switching*, a method of establishing communications that hails back to the earliest days of the telephone. 3G or Universal Mobile Telecommunications System (UMTS) was the start of the high-speed data connectivity. 4G networks, called Long-Term Evolution (LTE), brought a core network that relied entirely on Internet Protocol (IP) packet switching.

This last change brought mobile networks closer to the design of large-scale computer networks and the internet. Bridging the gap between Telecom networks infrastructure and the state of the art in computer networks is one of the core ideas behind the design of 5G. The 3rd Generation Partnership Project (3GPP), the organization that directs the specification of all new mobile networks, envisioned a new network with an ever widening range of use cases *i.e.* Machine to Machine

(M2M) for the industrial applications, automotive communications for smart cars, massive Internet of Things (IoT) communications for fleet of smart devices.

This ever growing list of new use cases comes with specific connectivity requirements. Receiving an update from a fleet of IoT does not require high throughput, but rather the ability to communicate with a large number of devices at the same time. For automotive needs, communication must occur with very tight latency constraints.

No single data connection can satisfy all these quality of service objectives at once. The main innovation of the 5G network is that it can serve different kinds of Service Level Objectives (SLOs) by relying on virtualization technologies and Software Defined Networking (SDN) to adapt to the needs of its users on the fly. Previous generations of mobile networks ran on a fleet of specialized and proprietary hardware, each dedicated to a specific network function. With this type of infrastructure, changing the capabilities of the network means that an MNO needs to send an engineer over to their data center who would then have to reorder and rewire the networking equipment according to its new purpose.

Network Function Virtualization (NFV) enables a new type of infrastructure where all network functions are Virtual Machines (VMs) that run on general-purpose hardware (or Commercial Off The Shelf (COTS)). 5G networks are intended to be able to be reconfigured through changes in configuration files, redeploying parts of the network by launching new VMs and managing interconnections between VMs through SDN.

Relying on a fleet of virtual machines is necessary to allow the network to be responsive, but the management of a virtual infrastructure is a new paradigm that is far different from what MNOs have been used to with the previous generations. These virtual functions will be lighter and more numerous than their predecessors. By studying the way they access, share and modify the network's data, we propose a set of consistency models that are more appropriate than ACID properties as to the resiliency requirements of the network.

Our study of the first official 5G specification and of the state of the art of consistency models in distributed systems, has led us to propose the following contributions:

- By extracting, grouping, and streamlining the relevant operations and data from the specification, we identify three crucial mobile network procedures.

We describe them in a simple format, enabling us to detect times when a data race could occur (Section 4.3).[1]

- By determining when and in which function each variable is modified, we bundle the contents of 5G's distributed data store in groups. For each group, we consider application-level invariants that should be respected to preserve the correctness of the network. We also propose consistency semantics that are sufficient to enforce these invariants while still maintaining availability (Section 4.4).

The consistency semantics we propose are sufficient to enforce 5G's invariants when only the data relevant to a single user is considered. In a situation where Network Slicing is used, the data of a slice can be distributed far more widely. [2] As a consequence of this, the enforcement of one of the invariants we present in Section 4.4.7 becomes more challenging. Namely, the enforcement of resource consumption limits for a distributed slice.

To provide tools for system designers that will allow them to consider the tradeoffs of using different consistency semantics to solve this resource usage limits, we propose the following contributions.

- An analysis showing that current rate limiting solutions in networks are not sufficient to solve this problem. And an abstraction over the network component designed to enforce network policies: the limiter (Section 5.1.1).

- A set of properties that can be used to assess the fitness of a given limiter abstraction when put against the requirements of a specific use case (Section 5.2).

- Four limiter designs placed at various points of the consistency spectrum, intended to illustrate the tradeoffs in the performance of these limiters.

- Our modifications to the EdgeCloudSim simulator to make it more suitable to the evaluation of limiters in a 5G network.

- Our assessment of the suitability of these designs to enforce a global limit, as well as the parameters that affect the performance of our designs.

[1] At the time of study, the 5G specification was still a work in progress for the 3GPP members. This combined with the primary goal of exhaustively describing every connectivity scenario foreseen by the authors makes for a chaotic reading experience.

[2] As we show it in Section 2.2.2, slices are shared among multiple users. The data related to a slice can consequently be much more distributed than other variables.

1.1 Structure of this manuscript

This document is structured as follows:

- Chapter 2 presents the two driving forces behind the design of 5G: Network Softwarization and Network Slicing.
- Chapter 3 presents consistency models that will be used in later parts of the thesis to reason on the consistency needs of the network.
- Chapter 4 presents an in-depth analysis of the 5G specification. We focus on three key network procedures and all the data that is manipulated by these procedures. We then state several invariants that should be sufficient to preserve the correctness of the system, and propose consistency mechanisms to enforce them.
- Chapter 5 presents the limiter, an abstraction that frames to the problem of enforcing global resources usage limits. As well as a set of properties to assess a limiter's fitness.
- Chapter 6 presents four designs for distributed limiters that follow our limiter abstraction.
- Chapter 7 presents SliceCap, the simulator in which we assess the performance of the limiter designs. We present how we modified the EdgeCloudSim [29] simulator and the results of our simulation.

5G Networks

2

This section presents the design principles behind 5G and introduces Network Softwarization and Slicing, the two driving forces of the 5G network.

A mobile network is composed of two parts. First the Radio Access Network (RAN), which evolves with new radio technologies for the mobile devices and in the towers that provide coverage. Second, the core network, the array of Network Functions (NFs) that connects the towers to each other and to other networks. The availability of new technologies and evolution in the way users want to use the network orient the design of each new version of these two elements.

Previous generations of mobile networks each brought their own evolution to both the RAN and the core network. We are only concerned with the evolution of the core network, thus the RAN is beyond the scope of our work.

For 5G, these evolutions are three new types of network connectivity and dynamicity: the ability to adapt itself to its users. Dynamicity has been made possible by the virtualization of the core network.

2.1 Network Softwarization

The logical components of the network are called Network Functions. Each function fulfills a specific role in the infrastructure.[1] 5G decouples the network architecture and the software implementing the NFs from its supporting physical hardware. Now these functions that used to be specific network appliances, with specialized hardware, run instead on top of COTS hardware.

Network Softwarization is the process of running NF in software instead of hardware. The hardware itself is abstracted away through the use of virtualization, and the interconnection of the VNFs will be adaptable through the use of SDN. All NF in 5G have been virtualized.[2]

[1] We present all the functions that are part of the 5G specification in Section 4.1.
[2] This virtualization processes itself is called Network Function Virtualization (NFV) [34].

With this virtual infrastructure Telecom Operators gain the ability to deploy and redeploy the core network as needed to serve their users. Through the use of orchestration software, this programmable network is able to adapt autonomously.

Dynamic slices, presented in Section 2.2, are an example of autonomously reconfiguring the network. Another example is the network topology reconfiguring in order to restore connectivity during a natural disaster [5].

The performance of a network is a consequence of the arrangement and the configuration of the VNFs that compose a service chain: a succession of User Plane Functions (UPFs) that link an UE to its target Data Network (DN).

As of now, three types of configurations have been standardized,[3] destined to serve three broad types of use cases by meeting their SLOs. Several service chains can coexist on the same hardware and support connection with far different performance.

4G networks also have the ability of supporting several connections with different SLOs. The 4G implementation is done through Quality of Service (QoS) indicators and traffic prioritization for certain emergency traffic. 5G's approach leverages virtualization, the use of orchestration, and cloud hosting through slices. Each slice is configured to fulfill the Service Level Agreement (SLA) of a specific use case. As a consequence, 5G is far more flexible, and opens the door to more use cases. New slices can be put into service with a configuration customized to the needs of a new use case.

2.2 Network Slicing

In addition to Network Softwarization, a major innovation of the 5G network is its ability to serve multiple use cases at the same time on the same hardware through Network Slicing. Network Slicing is a combination of several ideas that depends on Network Softwarization:

- Multi-tenancy: the ability to serve industrial clients while enforcing resources isolation between them. When the physical networks are shared among several clients, each of them will access a reserved portion of the available resources.[4]

[3] The three standard slices that have been identified are eMBB, URLLC and mMTC, shown in Section 2.2.1.

[4] A slice of the resources, if you will.

- On demand networks: the ability of the network operator to deploy resources *where* requested by the client, *when* needed if dynamic slicing is used.

  In some geographical areas of interest, industrial actors will be able to request the deployment of a static slice to serve their purpose. For instance, to communicate with equipment in their factories, or to access and control a wind farm.

  By relying on programmable networks, slices will be able to be deployed on demand when an authorized device request their use. For instance, with a priority slice that can be deployed to serve emergency services, even if the rest of the network is saturated with general-purpose users, they will use their reserved share of the network resources.

- Custom QoS: the ability to provide a type of connection that is appropriate with the client's use case. Network Slicing is the innovation that will allow the network to serve a range of connectivity requirements at the same time to different users[5].

  Different types of service can be offered thanks to specific hardware, scheduling rules, routing algorithms, prioritized packets, or a combination of all the former techniques, and other network resources.

2.2.1 5G standard slices

5G identifies three standard slices that can serve a broad range of use cases.

- Enhanced Mobile BroadBand (eMBB): This is an enhanced form of the connectivity offered to LTE users. General-purpose data connectivity with marked performance improvements. Especially in terms of high density uses *i.e.* being able to connect despite being in a crowd, in a sports stadium. Or in terms of increased peak throughput.

- Massive Machine-Type Communications (mMTC): serves device communications on a massive scale. This will allow users to pilot a large park of IoT devices. For instance, in logistical applications, where every device must to report at the same time.

- Ultra-Reliable and Low-Latency Communications (URLLC): serves industrial use cases; through reliable and latency bound connectivity. *e.g.* vehicular connectivity or control of industrial equipment.

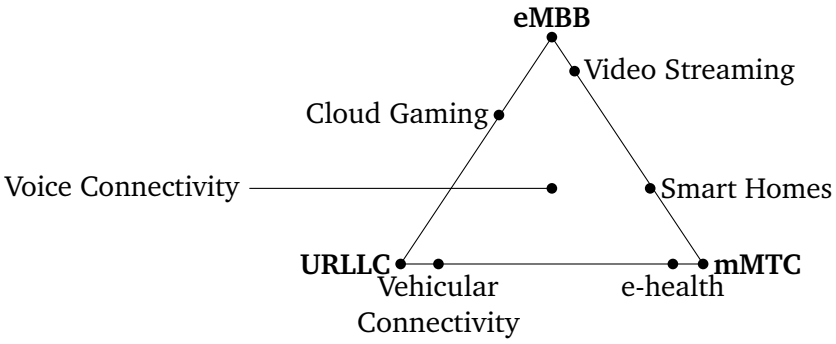

Fig. 2.1.: The three 5G service types and use cases.

For use cases whose requirements are not met by the standard slices, a custom slice can be deployed by the Telecom Operator. The SLA of these custom slices can differ from the standard ones, but cannot cover all of them. That is because the requirements of the three standard types of service are incompatible. The network cannot serve packets at peak throughput and with as little latency as achievable at the same time. A choice must be made according to the use case that is targeted.

As Figure 2.1 illustrates different use cases can be placed on an axis according to their requirements. For instance, vehicular connectivity is highly latency bound, but can benefit from mMTC connectivity. Video streaming has to be broadcast to multiple users and use as much bandwidth as available.

Generally speaking, use cases leveraging 4G's bandwidth and peak rates would naturally transition to an eMBB slice in 5G. This is why this slice will be the most commonly deployed by network operators to support the most common use cases. The other slices complete the network operator's service offer to address industrial usages that include IoT type applications, factory automation and more.

In contrast to these new use cases, voice connectivity has no requirements on latency, broadband or reliability that were not met by previous generation networks. It is perfectly able to run on existing, general-purpose networks (as it does to this day).

2.2.2 Distribution of slicing related data

Slices can be deployed over a large area and they serve a set of users, instead of a single subscriber. For slices that serve a specific use case, all the users that

are allowed to connect share a unique set of resources that they are allowed to consume.

As part of the state of every slice, there needs to be a user-centric approach that quantifies the resources currently used, and the user's spending limit in this slice. For the general-purpose slice, this data is specific to the user and correspond to its data plan. It must remain consistent to avoid cases where the user's usage exceeds its current offering. For a slice serving an industrial use case, this state is common to all users of the slice, and it must remain globally consistent in the network.

2.3 5G scenarios

The variety of use cases that are made possible by 5G is great, especially because Network Softwarization makes the network more adaptable to unforeseen use cases. We propose to illustrate this through three scenarios that are made possible by slicing. These scenarios are working hypotheses and show how industrial clients will be able to use 5G. Each comes with different locality constraints and connectivity requirements.

2.3.1 Itinerant Slice

In this first scenario, we consider the needs of a TV show where the filming location changes regularly. Every few weeks for instance. At each new location, the filming crew sets up audiovisual equipment and during the show a video stream is broadcast live to the audience.

In this case, the show producers have to be able to rely on the upstream bandwidth. The general purpose eMBB is often asymmetric because users consume content instead of creating it, and their upstream and latency expectations are less stringent. They are more likely to upload a file rather than broadcast a live video.

Moreover, when part of the audience can be present during the show, there could be parts of the audience who would use their own mobile device. This could monopolize some of the bandwidth that is necessary for the show to take place without incident. That is why the producers need to reserve some bandwidth.

In short, this use case requires some reserved network bandwidth, with specific QoS geared towards video broadcast. This connectivity needs to be deployed in one specific geographical location, somewhere in the area covered by the Public Land

Mobile Network (PLMN). While this location is bound to change, it is not a frequent event.

This use case can be covered with a dynamic slice that is deployed at the filming crew's current location. This deployment happens when the network detects that a device is attempting to establish de data session over the slice.

2.3.2 Industrial Slice

For this scenario, the client is an industrial actor with several factories and general offices. First they require guarantees on the interconnection between their different locations. In a manner that is similar to a Virtual Private Network (VPN), but with additional connectivity guarantees that only the network operator can provide.

In addition to this, the factory floors are filled with devices that communicate with each other and their controllers through M2M communications. In some location, the nearest PLMN antenna is used to support this. In others, the client has purchased its own antennas for this purpose. Either because of regulatory constraints or because they lack this expertise in-house, the antennas are piloted by the MNO with its own control plane.

This use case, one static slice can be deployed over several locations at the same time. The operator allocates some resources to provide a stable connection between the different locations. For all shared resources, slicing is used to reserve some capacity and to provide isolation from other users of the network. The antennas that belong to the client are reserved for its exclusive use.

2.3.3 Critical Slice

In this scenario, we consider a metropolitan area with several hospitals with urgent care services, as well as a fleet of ambulances.

When a patient is on board an ambulance, the crew monitors their vitals by use of connected devices. This data, as well as their observations, are uploaded on a medical network shared by the hospitals and the ambulances. This network is used to automate and streamline triage.

All communication over this network is given absolute priority. While resources are reserved statically around the hospitals, they also follow along the path of any ambulance that holds a patient on board.

This use case can be covered by one slice with as many static deployments as there are hospitals, and as many dynamic deployments as there are ambulances trying to connect to the network. Wherever this slice is deployed, the operator must guarantee that resources are made available for it. That is despite the operator or the client being able to predict where uses will occur.

3 Consistency

This section presents consistency models and the CAP theorem. In Section 5 we explore the consistency challenge of limiting the global resource consumption for slices.

The 5G network is a stateful distributed application. This state is distributed among the NFs that are the components of the network and the Unified Data Management (UDM) that serves as the central data store of the system[1]. To behave correctly, a distributed application must know whether it can trust the state it holds to be correct.

How much confidence can be put into the data store depends on its consistency model. This section is an overview of consistency models. Our aim is being concise and easily understandable rather than presenting an exhaustive survey of all existing consistency mechanisms. We intend to present the consistency models that we will rely on for our analysis of the requirements of the 5G network procedures in Section 4.4.

The consistency model chosen for the UDM in the specification has been proven impossible by the CAP theorem, and that this level of consistency is stronger than required to maintain the correctness of 5G. Section 3.2 presents this impossibility result of the CAP theorem. It explains why a strong consistency model is incompatible with a resilient 5G network which motivates our study of weaker consistency models applied to 5G. Section 4.4 will present weaker consistency models that can be sufficient.

3.1 Definitions

This chapter is intended to present different consistency models for the data held in 5G. To do so, we need to define a few elements first.

[1] What that state consists of is the subject of Section 4.

The network relies on data to function. The UDM will serve as the network's data layer: it's central database. The scale of 5G means that this database will be distributed.

In our model, this database is a key/value store[2]. Clients use this data through read and write operations. Each client is a process: a sequence of operations on the values. For the network to behave correctly, the values returned by the database must make sense according to the value that was written into it.

We say that there is an anomaly when the data does not make sense, or when it violated a constraint. Examples of anomalies include

- Reading a value that was never written.
- Reading a value that was partially written.
- Never reading a value that has been written.

Different consistency models admit different anomalies. The literature is rich with consistency models, and they can be partially ordered by strength. A consistency model is weaker than another if it admits a sequence of operations with more anomalies. However, two models cannot be compared if the two sets of anomalies do not intersect.

It is in the nature of distributed system to be subject to network partition. When nodes or sections of the system are isolated from the rest due to inevitable failures. Partition tolerance denotes the ability of the system to keep operating despite these failures.

Our data is held in a distributed Key/Value store. This store maps a set of unique keys to an opaque value. A set P of N processes $P = \{P_1, \ldots, P_N\}$ operates on this store through read and write operations. We note these operations σ. The data store propagates operations between replicas. An operation that originates from another replica is either an update or a query. A write noted $w(x)v$ assigns value v to key x. A read noted $r(x)v$ reads value v at key x.

We note L_i the local history of P_i. The local history is the order in which each process sees its own operations (sometimes called program order). We note that σ_a and σ_b are ordered in L_i, as $\sigma_a \underset{i}{\rightarrow} \sigma_b$

Finally, the collection of all local histories is the history H of our system: $H = (L_0, \ldots, L_N)$.

[2] A relational database would require additional guarantees, such as referential integrity.

A consistency model determines what histories of read and write are legal with respects to some invariant, as we describe later. [3]

There are three kinds of consistency models: *a single-object model* make guarantees with respect to a single object only. *A multi-object* model makes guarantees across several objects. A *transactional consistency model* is a multi-object model that guarantees that a transaction (a delimited set of operations) is made visible atomically (all or nothing).

3.1.1 Sequential Consistency

Sequential Consistency is a single-object consistency model. It has been defined intuitively by Lamport as: "*the result of any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program*". [18]

The system behaves as though a single copy of the data existed somewhere. And all operations were ordered by their arrival in this location. This means that there must be a serialization of all operations within H that respects the following constraints:

- It maintains a single sequential order among all operations from all processes.
- This sequential order respects the program order within every local history L_i.

3.1.2 Linearizability

Linearizability is also a single-object consistency model. A Linearizable system is Sequentially consistent and respects real time. It builds on top of Sequential Consistency, with the additional ordering requirement that real time be respected. All operations appear to take place atomically at one specific point in time. This point of linearization appears to take effect after the start of the operation and before its return. All linearization points are totally ordered, and this total order respects real time[18].

If σ_a finished before the start of σ_b, then σ_a must precede σ_b in the total order.

[3] We call them invariants because they must always hold true.

3.1.3 Snapshot Isolation

Snapshot Isolation is a transactional and multi-object consistency model[10]. It involves a set of operations σ performed in order and potentially on multiple keys.

In this model, each transaction appears to work on an independent snapshot of the database. No changes are visible to operations outside the transaction until it has been committed. After commit time, all changes made within the transaction are visible to all future transactions.

Snapshot isolation introduces the notion of *visibility*. No changes are *visible* to any other transaction before commit time. Changes are made visible atomically: all changes are visible at once unless the transaction aborts. Then no changes will be made visible.

3.1.4 Serializability

Serializable consistency is a transactional and multi-object consistency model. Intuitively, it means that transactions are made visible in a total order. Serializability implies Snapshot Isolation and some other guarantees that we have not presented.

Because Serializability implies a total order, all processes see all operations in the system in the same order. And operations are applied atomically to potentially multiple keys through transactions. This implies that there is no interleaving of operations between transactions.

3.1.5 Strict Serializable

Strict Serializability is the strongest consistency model. It is a transactional and multi-object model that strengthens of both Serializability and Linearizability.

All operations are made visible in the same order for all processes. And this total order is consistent with real-time ordering.

3.1.6 Causal Consistency

Causal consistency is a multi-object model. It imposes a partial order. Each process may perceive a different serialization, but they must respect some ordering guarantees. Intuitively, Causal Consistency tries to introduce a logical relationship between operations in H. To explain Causal Consistency, we must introduce two new relations:

Write Into $\mapsto$

Write into $\mapsto$ define a dependency between write operations and read operations[3]. Intuitively, if you read a value in σ_i, it must have been written at some point. Ahamad et al. define these relations as follows.

- All reads follow a write: If $\sigma_i \mapsto \sigma_j$ then there must be a key x and a value v such that σ_i = w(x)v and σ_j = r(x)v.
- A read may only depend on one write. For any operation σ_j there is at most one operation σ_i such that: $\sigma_i \mapsto \sigma_j$
- If a read follows no write on a value, the value returned is the initial value: If for $\sigma_j = r(x)v$ there is no σ_j such that $\sigma_j \mapsto \sigma_i$, then $v = \perp$.

Perrin et al. has extended this definition with the more general case of reads being dependent on more than a single write operation: *The definition of writes-into orders is deeply related to the semantics of registers. For other abstract data types, e.g. graphs, counters or stacks, the value returned by a query does not depend on one particular update, but on all the updates that happened before[23].*

With a more complex data type such as a Conflict-free Replicated Data Type (CRDT), a read may depend on writes from two different replicas. When two updates are concurrent, they are in conflict and can be merged. This merge operation is dependent on as many write operations as there are conflicting versions. We do not explore this more general definition in this work.

Happened-before relationship $\rightsquigarrow$

This has been defined by Lamport [20] as follows.

- $\sigma_i \rightsquigarrow \sigma_j$ if $\sigma_i \rightarrow \sigma_j$. If a process P_i sees two operations ordered in its L_i. Then these operations are linked with the happened-before relationship.

- $\sigma_i \rightsquigarrow \sigma_j$ if $\sigma_i \mapsto \sigma_j$. If σ_j reads a value written by σ_i, then σ_i happened-before σ_j.

- $\sigma_i \rightsquigarrow \sigma_j$ if $\sigma_i \rightsquigarrow \sigma_k$ and $\sigma_k \rightsquigarrow \sigma_j$. Happened before is a transitive relationship.

If two operations σ_i and σj are not linked by the happened-before relationship, then this means that they are not causally related. These two operations are concurrent.

Writes that are potentially causally related must be seen by all processes in the same order. Concurrent writes may be seen in a different order on different machines[30].

Causal Order

Causal Consistency applies over the local history of each process in the system. They may all see a different ordering of all operations in H, but this order must respect the causal order of all operations. The causal order is the transitive closure of the union of program order $\rightarrow$ and $\mapsto$, as shown above.

The intuition of Causal Consistency is that where Linearizability tries to impose a total order linked with real time, Causal Consistency tries to impose a partial order linked with logical time. Program order imposes an order dependency that must be respected. And a data dependency is also introduced with the *write into* relationship.

Causal Consistency must be built on top of four session guarantees.

A session is an ordered set of operations. The signification of the term session is loosely defined and depends largely on the system or the application being used. Here we understand session as the succession of operations as observed by a process. Similarly to L_i These definitions are inspired by Viotti and Vukolić [35].

Read your writes

If a process P applies a write operation $\sigma_i = w(x)v$, then all subsequent reads in this session must read this value $\sigma_{i+n} = r(x)v$

Monotonic reads

Let there be two operations applied by a process P: $\sigma_i = w(x)u$ and $\sigma_j = w(x)v$. Once any read operation has returned $\sigma = r(x)v$, then no other operation may return $\sigma = r(x)u$

Intuitively this means that our data stores move only forward in time. It cannot return previous values.

Monotonic writes

All writes belonging to the same session must be applied in the same order.

Write follow read

Also called session causality. All writes operations made during the session are ordered after any writes in any session whose effect were seen by read operations in this session.

If $\sigma_j = r(x)v$ is a read operation made during the current session. And $\sigma_i = w(x)v$ is a write operation on the same key made during another session. σ_i reads the value written by σ_j : $\sigma_i \mapsto \sigma_j$, then if any write is made afterwards during the session $\sigma_k = w(y)u$, σ_i must always be applied before σ_k.

3.1.7 Eventual Consistency

Eventual consistency is a model that does not provide ordering guarantees for operations. Instead it offers a guarantee on the state of a replica. This state is the result of the application of all operations. In systems that support this model, and in the absence of new updates, replicas will eventually converge to an identical state[24].

The convergence definition of Eventual Consistency is that *Correct replicas that have delivered the same updates eventually reach equivalent state*[26].

3.1.8 Strong Eventual Consistency

Strong Eventual consistency is a guarantee beyond having all replicas converge to the same state. Under SEC, all replicas eventually reach some deterministic state.

The convergence definition of SEC is *Correct replicas that have delivered the same updates have equivalent state*[26].

To achieve this, system designers rely on specific data structures to achieve this, instead of relying on the order of updates. Examples of the specific guarantee include Operational Transformations and CRDTs.

3.1.9 Causal+ Consistency

Causal+ consistency[21] also add a convergence requirement to Causal Consistency. This means that all correct replicas that have applied the same update hold the same state, like in Strong Eventual Consistency.

This requirement can be fulfilled by relying on data structures that provide a deterministic and convergent merge, such as CRDT.

3.1.10 Transactional Causal+ Consistency

Akkoorath et al. [4] propose Transactional Causal+ Consistency. Snapshot isolation provides the isolation needed for transactions, and the use of CRDT to enforce the strong convergence requirement.

3.1.11 Tracking Causal Dependencies

When a distributed data store guarantees causal consistency, the order in which operations are applied must respect all the guarantees we presenter in Section 3.1.6. The data store has to guarantee that all dependencies are satisfied for every operation it receives from a replica before it applies it locally. For instance, the monotonic writes guarantee requires the data store to know if all writes have already been applied for the current session.

If the data store receives an operation σ_j and $\sigma_i \rightsquigarrow \sigma_j$, the data store must detect that no operation σ_k exists such that $\sigma_i \rightsquigarrow \sigma_k \rightsquigarrow \sigma_j$.

This is done by tracking the causal dependencies of all operations, and we present a few methods used to do that here. For all these tracking methods, there is a tension between *freshness*: being able to apply updates as soon as possible, and the size of the metadata and the complexity of the dependencies system.

Using Scalar Metadata

In GentleRain[15], a local operation is applied immediately to its replica. It has a scalar timestamp is derived from the physical clock. When these modifications are propagated to the other replicas, when the system transmits an update, it piggy-backs the corresponding timestamp as metadata.

GentleRain stores all the updates received, but they are not visible at once. Instead GentleRain computes a Global Stable Time that is the minimum of the greatest clock it has received from any of the replicas. An update whose clock is smaller than the GST is made visible.

This metadata is small and does not add a lot of overhead to the propagation of updates. However it comes with a problem of over-approximation: updates that are not causally dependent on each other can still be delayed before being made visible.

This also means that when a single replica is unavailable, the system cannot progress. No updates can be made visible.

Using a Vector Clock as Metadata

In Cure[4], local operations are still visible immediately but the metadata that piggy-backs the updates takes the form of a vector clock instead of a scalar. This vector contains the greatest clock for each replica that was visible when the operation was made.

This tracking is more fine-grained and limits the problem of spurious dependencies. Two replicas can still exchange each other's updates, even if they are partitioned away from all other replicas. However this means that the overhead of updates grows with the number of replicas.

Tracking Causal Dependencies Through Causal Channels

Saturn[11] track causality dependencies with labels. The metadata of updates contains this label, and labels are also exchanged among replicas through a causal channel.

These causal channels are arranged in a spanning tree topology. This means that each replica needs to know every neighbor and the total network topology. The labels are of constant size, and they offer more precision than vector clock. If a replica becomes unavailable, the tree will need to be rebuilt, and a vector clock serves as a fallback.

Reduced Spanning Tree

ChainReaction[6] reduces this spanning tree to a chain. Write operations are submitted to the head of the chain and return once they have been applied to k replicas. Once a write has reached the tail, it is considered stable. Since the head of the chain is the replica where all writes are submitted, all dependencies are already satisfied when a client submits a write operation.

This does mean that a single replica is the bottleneck for all operations. But this load can be balanced by sharding the key space and having several chains for each shard.

3.2 CAP Theorem

Out of all the consistency mechanisms we described in Section 3, only some of them are achievable by a distributed system while still remaining available. Consistency mechanisms can be described in terms of strength. Linearizability is stronger than Sequential consistency. Eventual and Causal consistency are both weak consistency models.

Brewer's CAP theorem[16] states that a distributed system in the presence of network partitions (P) must choose between (C) strong consistency and (A) availability. In the sense of CAP, consistency stands for strong consistency semantics like Serializability.

The intuition of this impossibility result is that to remain strongly consistent, a data store needs to synchronize its replicas when running an operation. With

no synchronization, the system cannot guarantee that all invariants remain true globally.

With Eventual consistency, the invariants do not decree that replicas cannot diverge from each other during a network partition. Only that all replicas will converge again after resuming normal operations.

With Causal consistency, the invariants decree that an operation cannot be applied until its dependencies have been satisfied, in order to maintain the logical dependencies of all operations. This can differ according to the means of tracking causal dependencies (see Section 3.1.11). Usually local operations can still proceed, because their logical dependencies are satisfied by the current state of the replica. It is the updates being received that may not be made visible until normal operations resume.

However strong consistency models require the participation and synchronization of multiple replicas for each new operation. For instance, with a quorum being necessary to validate a modification. This means that during a network partition, no isolated replica may run an operation for a client. The replica will become unavailable in order to preserve the consistency of the data.

It is for all these reasons that highly available data stores cannot provide Atomicity, Consistency, Isolation, and Durability (ACID) guarantees. Conversely ACID data stores cannot be highly available.

Causal consistency, or more precisely Transactional Causal+ Consistency (TCC) is the strongest consistency model that is still available under CAP[31].

There are two incomparable[4] models of consistency that are considered to be the strongest model that remain available under CAP. Transactional Causal+ Consistency[31] and Monotonic Prefix Consistency[17].

3.2.1 Hybrid Consistency

Some data stores offer consistency semantics that are not applied globally. Instead consistency guarantees are applied according to the purpose of the data.

[4]These two models are incomparable, because there are linearizations that MPC allows and CC forbids, and vice versa. MPC requires all replicas to agree on a total order for part of the operations (the *prefix*), which Causal Consistency does not. MPC does not require this order to respect causality, conversely MPC expose processes to arbitrary rollbacks.

AccGreGate[36] is an example of a data store that provides a hybrid level of consistency. In this system, there are two datasets. One contains media: such as posts or pictures. The other contains access rights. This models social media, with access rights being the list of friends. The access rights dataset has stronger consistency semantics, with all operations being ordered in relation to each other. The media data type is more relaxed. But all media operations depend on one access right operation. This enables media posts to be visible more quickly while stile retaining the correctness of the access system.

TiDB[19] is a data store that provides hybrid replicas. There are two kinds of replicas: one is a key/value store that provides a lower level of consistency than the second type of replica. This second type of replica is an SQL database that allows users to run complex queries for analytics. SQL data stores rely on the referential integrity provided by ACID. (Intuitively referential integrity means that the existence of a reference guarantees that the data being referenced also exists.)

3.2.2 High Consistency Zones

Ojala et al. in their study of 4G networks propose Highly Available and Consistent (HAC) zones[22]. In their work, the data stored by the 4G core network is distributed over a global Eventually consistent data store. The data stored in this database is the User Equipment Context (UE-context): a set of values that concerns one specific device being used to access the network. Because this one specific device is located in a geographical zone, this data is mostly used in a specific neighborhood.

This solution relies on the physical proximity of all the replicas to circumvent the CAP theorem. This is possible because at this scale latencies are far lower. And also because the network is more reliable locally. When the device moves, this HAC zone moves with it. With replicas entering and leaving the zone.

Operations will eventually be replicated to the rest of the network. But if the data in a replica far away from the UE has not yet converged and is still out of date, the correctness of the network will not be compromised.

An example of this is the location information of an UE: if a message must be routed over to the UE, then even if the location data is out of date, as the message travels closer to the device, the replicas that hold this location data will have received more updates. The data will become more consistent as the distance with the device decrease. That is until the message enter the HAC zone, where the location data will be the most up to date.

4 Data Lifecycle in the 5G Specification

In this chapter, we propose an analysis of the main procedures of the mobile network and the data stored in the UE-context. The UE-context contains the data that concerns a device used by a subscriber to connect to the network. Our analysis of the 5G specification in this section will help link each variable in the UE-context to the procedures as well as the VNFs that access/modify it.

Our work is based on the first official 5G specification (*i.e.* 3GPP release 16 [1, 2]). This document specifies the UDM, the VNF tasked with the storage of most[1] of the UE-context. Within the specification, the UDM is described as a distributed, transactional database with full ACID guarantees. Brewer's CAP[16] theorem proves that a distributed system cannot be strongly consistent and resilient against network partitions (see Section 3.2). With these guarantees, a transactional system will not be able to function correctly during network partitions.

Telecom Operators focus on preventing network partitions in their infrastructure through network and components redundancy, for example. However it is inevitable that partitions will occur. Considering the CAP theorem, 5G cannot rely on a strongly consistent database, or it risks becoming unavailable during partitions. *That is why it is important to design a system able to remain correct during partitions without becoming unavailable.*

Correctness is a set of requirements that is different for every application. This is why we trace the life cycle of data in this chapter. By tracing the usage patterns, we want to be able to outline the consistency requirements of the system. We are able to show that because of how often data used in 5G is used within close proximity of the UE, the consistency mechanisms necessary to maintain correctness are actually far weaker than the level specified for most of the 5G data.

Related work includes Ojala et al. [22] who proposed a distributed core network where a distributed NoSQL database stores the UE-context. In a 5G system, Slicing related data provides an exception to this that we will explore in Section 7.

[1] With the exception of all authentication-related data, which is managed by the Authentication Server Function (AUSF), defined in Section 4.1.1

- Section 4.1 presents the architecture of the 5G Core Network and defines its components.
- Section 4.2 presents how we will classify variables according to *how* and *when* they are used by the network.
- Section 4.3 proposes three mobile network procedures we extracted from the specification. We analyze these procedures in detail, the variable that they handle, and the interplay between variables and Network Functions.
- Section 4.4 links all the variables to the modifications classes defined earlier. We present the variables in batches according to these patterns and define necessary invariants for the set of variables. We also link to consistency mechanisms presented in Section 3.

4.1 A presentation of the 5G architecture and its components

In this section, we present the architecture of the 5G core network, as well as the components of the core network: the VNFs. The VNFs presented here are the entities that will exchange messages during the procedures in Section 4.3.

The purpose of the 5G core network is to manage User Equipments (devices) called UE, allow them to connect to their target Packet Data Network (PDN) according to the operator's policies, and to charge subscribers for their use of the network.

The line dividing Figure 4.1 in two parts separates the control plane, above the line, from the data plane below. The Network Functions in the data plane are tasked with carrying data between the UE and DN, using the radio technologies of the RAN. The control plane is tasked with the setup of these connections, and will be the main focus of our work. This data plane as well as the RAN in charge of the radio technologies are beyond the scope of this paper.

4.1.1 5G Network Functions

Figure 4.1 illustrates the components of the 5G network. These components shown as rectangles on the figure are Network Functions. A network function is a building block of the network. Every network function exists to fulfill one specific role in the

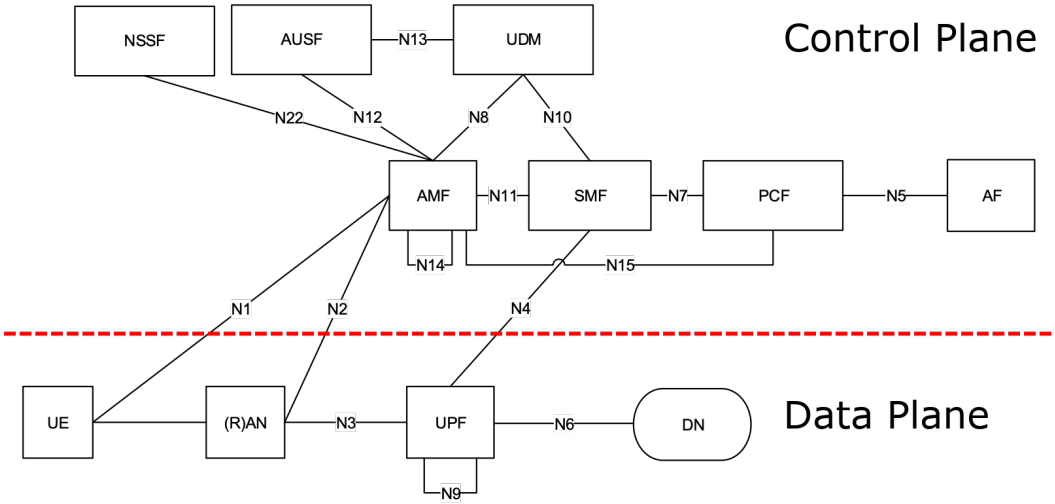

Fig. 4.1.: 5G System Architecture [1]

architecture. In 5G networks, NFs are no longer specific devices, but rather software running on COTS computers: they have been virtualized, and so are called VNF.

Functions involved in the procedures

- User Equipment (UE): This is the function embedded in the device that allows the subscriber to access the network.
- Access and Mobility Function (AMF): The is the central function that manages the UE. It receives requests directly from the UE and makes decisions accordingly or forwards them to different VNFs
- Authentication Server Function (AUSF): This is a server that authenticates the UE when it first connects to the network. It maitains security information in its own section of the UDM and installs a security context on the Access and Mobility Function (AMF)s and the UE so that all future communications with the network are encrypted.
- Session Management Function (SMF): This is the VNF in charge of setting up and managing one or several Packet Data Unit Session (PDU-Session). It will not change during the lifetime of the session.
- Unified Data Management (UDM): This is the unified interface to 5G's data store. It allows 5G core network components to query, subscribe to, and modify values.

- Policy Control Function (PCF): This VNF ensures that all policies (QoS, Spending Limits, allowed types of connection) regarding a subscriber, an UE or a PDU Session are known and kept up to date.

- User Plane Function (UPF): This is the component that carries data packets between the UE and the PDN, either directly or through Intermediate UPF (I-UPF). The PDU Session Anchor (PSA) is the closest to the PDN and is chosen for the lifetime of the session.

Additional Network Functions

For the sake of completeness, we describe the functions that will not be directly involved in the network's procedures we describe.

- Application Function (AF): These are application specific functions running within the 5G infrstructure. This allows the to interact with the Policy Control Function (PCF) or even to influence routing decisions.

- Network Slice Selection Function (NSSF): The Network Slice Selection Function (NSSF) is a repository of information on slices. It is intended to advertise their features when several are availble for the user.

- Radio Access Network (RAN): This function handles the radio access technology and the communication between the UE and the network.

- Data Network (DN): The Data Network (DN) or Packet Data Network (PDN) is the network targeted by the UE, for data or voice services, the Internet for instance.

4.2 Classifying Variables Modifications

There is a lot of data in the network. And to be able to reason about it more simply we have chosen two ways of considering these variables. We begin with *when* the data is modified by showing the different categories of triggers in Section 4.2.1. We then show *where* data is used or modified in the network in Section 4.2.2.

4.2.1 Data Modification Triggers

The 5G system uses and stores many variables in order to provide connectivity to user devices. But there are fewer triggers for data modification. Here is how we classify changes in the data.

- Changes triggered by the UE.

  When the UE's location changes. This is the most obvious change, as expected for a *mobile* network, and a procedure is dedicated to handling this event. For instance, it triggers changes in the data used to identify the AMF[2] responsible for the UE.

- Periodic changes coming from the parts of the system we consider.

  A data session may be set up for the subscriber. Every subscriber has a Spending Limit, and this value has to be periodically updated to reflect the subscriber's usage.

- Changes coming from outside the part of the system we consider. These are changes that have been made manually to the data.

  The Steering of Roaming[3] is data that can be modified at the discretion of the Telecom Operator. But this is a matter that is beyond the parts of 5G we consider and most likely subject to complex negotiations between Telecom Operators. We consider it as constant in the system.

4.2.2 Data Modification Locations

We can now classify all access to the data by determining when a change can occur. There is a second matter that we have to consider: the locality of changes in data. As we mentioned in Section 3.2.2 with Highly Available and Consistent zones, the locality of data must be taken into account to deduce what consistency models will be achievable.

[2]The AMF will be defined more precisely in Section 4.1.1. Here it can be considered as being the antenna.

[3]This is data that describes how the UE can connect to a network from a location that is beyond the coverage of the Subscriber's Telecom Operator.

- *Local data*: variables that are always used by operations in a specific geographical zone. They never leave their geographical location. For instance, *data plane variables*: the tunneling information. Only User Plane Function (UPF)s need them to connect to each other, and the UE to make use of the tunnel.

- *UE specific data*: variables that belong in the UE Context but are modified by operations from a limited number of VNFs. e.g. when the UE changes locations, a new AMF will overwrite the previous one's identifier in the UDM.

- *Distributed data*: variables that are inherently more distributed: A greater number of VNFs modify their own version independently. e.g. the value of the spending limit for a subscriber is the result of every modification from any PCF associated with a Packet Data Unit (PDU) session belonging to the subscriber. And this in turn depends on the reports made to all associated Session Management Function (SMF)s.

4.3 5G Network Procedures

Now that we have presented the elements of the 5G network, we can show how they interact when serve users. We have chosen three procedures that are at the core of the functionalities of the network.

(i) *Registration*: When a device first connects to the network and its UE-context is queried from the UDM. (Figure 4.2)

(ii) *Packet Data Unit Session (PDU-Session) Establishment*: When a device initiates a data connection. (Figure 4.3)

(iii) *Handover*: When a device has moved and must be recognized by the network at its new location and have its PDU-Session migrated when it tries to use them. (Figure 4.4)

The specification describes these procedures at length. While the specification attempts to list all possible scenarios (a registration request to establish Roaming connectivity, the user trying to make an emergency call, a handover between antennas not supporting the same generation of mobile networks, etc.) we have streamlined the procedures by limiting them to the most common use case for a fully deployed 5G network.

4.3.1 Contents of the UE Context

This section presents all the 5G variables that are used in the network procedures that we present. We have extended our approach of simplifying the network procedures as far as possible to the data exchanged within them. We have removed all variables that were only of use in very specific use cases. We also merged together tightly linked variables that were only used or modified together. Our goal was to make this list simple enough to be understandable by readers unfamiliar with 5G, and familiar enough to be recognized by readers more familiar with Telecom's parlance.

- Data Network Name (DNN): This is an identifier for the Data Network the subscriber is trying to reach.
- Local Area Data Network (LADN): A Data Network Name (DNN) that may only be accessed while in physical proximity. Every AMF knows all the nearby Local Area Data Network (LADN)s. [4]
- Multimedia Priority Service, Mission Critical Service (MCX): These are information concerning the degree of priority that this UE should be afforded.
- Mobility Restrictions: A set of restrictions applied to the UE. They restrict mobility handling procedues and service access (e.g. access closed to groups, forbidden areas).
- Packet Data Network (PDN): Describes a Data Network where pakets are used instead of circuit switching. Most common examples are IP networks.
- Packet Data Unit Session (PDU-Session): A PDU is a logical connection set up between the UE and a target PDN.
- Packet Data Unit (PDU): Packet Data Unit (PDU) a unit of information that is exchanged between the UE and any PDN that the user is subscribed to (i.e. IPv4, IPv6, Ethernet). PDU connectivity is the service that provides the exchange of PDUs between the UE and a PDN identified by its DNN by establishing a *PDU Session*.[1]
- Permanent Equipment Identifier (PEI): A unique identifier for the device through which the UE is reaching the network *e.g.* the IMEI
- Priority Flag: This is a flag set in the header of the session establishment messages when appropriate according to the MCX priorities.

[4] This is how the 5G network plans on implementing the Mobile Edge Computing (MEC) model.[14]

- Quality of Service (QoS): This contains all the information on the performance of the connection.
- Single NSSAI (S-NSSAI): This is the identifier of a slice selected through the Network Slice Selection Assistance Information (NSSAI)
- Subscriber Data Management (SDM): This is all the information pertaining to a Subscriber.
- Session Management Context: This contains all the information pertaining to current PDU sessions and to the set up of future sessions.
- Subscription Concealed Identifier (SUCI): This is a single use identifier for the client that can be broadcast in clear to the AMF while still preserving the anonimity of the subscriber.
- Slicing Change Indicator: When this value changes it signals to the VNF that the subscription data for network slicing changed and the UE configuration must be updated.
- Steering of Roaming: The information used by the UE to know which foreign networks it should connect to while abroad.
- Tunnel Info: Contains the information necessary to route packets to the correct UPF, PDN and UE.

4.3.2 Diagrams Presentation

Figures 4.2, 4.3, and 4.4 each represents a network procedure. This section is intended to present the notation that we used within the diagrams.

Each column represents a VNF. Times pass from top to bottom. The arrows linking VNFs together are messages being exchanged during the procedure. Each arrow is named after the step in the procedure it represents. When a message carries data, the list of variables is represented between brackets. Finally, the vertical boxes represent a step during the procedure that only involves a single VNF.

Figures 4.3 and 4.4 are further separated into two sections. Figure 4.3 shows a PDU-Session establishment as well as a Policy Change. Figure 4.4 shows a Handover operation as well as a Service Request. Within the specification, these are presented as four distinct network procedures. We believe that presenting them together simplifies the understanding of each.

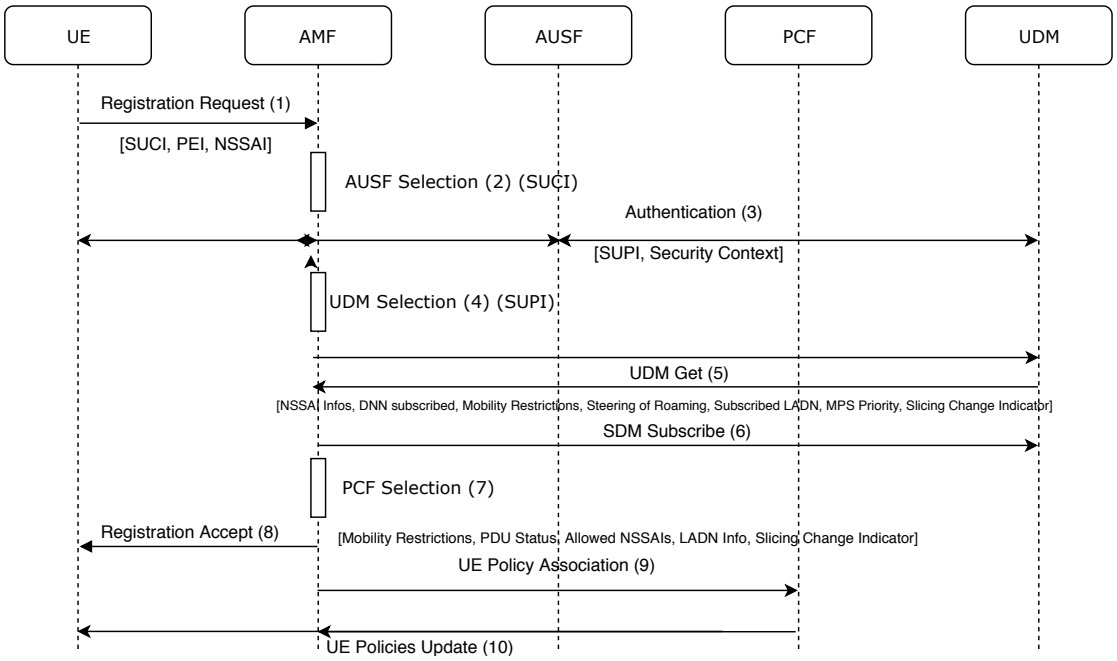

Fig. 4.2.: UE Registration Procedure[2]

4.3.3 Registration Procedure

The *first registration* procedure is used when a UE joins the network for the first time. A migration to another part of the network would be considered a *handover* or a *mobility registration*. The following steps are illustrated in Figure 2.

This section shows how identifying data is used when an UE first joins the network. And how a copy of the UE-context and Slicing data are first obtained and subscribed to.

(1): The UE sends identification data to the AMF. The SUCI will be the only identifier that is sent without first being encrypted, and that is used to obtain the Subscription Permanent Identifier (SUPI).

(2): The AMF uses the SUCI as an index to select the AUSF that is responsible for this Subscriber.

(3): The AUSF requests the authentication data concerning this subscriber from the UDM. It authenticates the UE and, if successful, installs a security context on the AMF and the UE. This Security Context allows encryption for all further communication between the UE and the network. The AUSF also provides the Subscriber's SUPI to the AMF.

(4): Using this new identifier (the SUPI), the AMF is able to select an UDM that is responsible for the UE context.

(5): The AMF contacts the UDM and requests the information necessary to set up a local copy of the UE Context. Some of this information will be sent over to the UE in (8). The Slicing Change Indicator will allow the UDM to inform its subscribers and the UE when their version of the data is out of date.

(6) The AMF subscribes to all future changes to the data it just obtained. Some of this data is likely to be updated and it is important for the system to ensure that all the VNFs see an up-to-date version.

(7) The AMF selects a PCF responsible for the UE It will be called upon later. (See 9, 10)

(8) The AMF returns some of the UE Context it just pulled from the UDM to the UE along with an acceptance of its registration. At this point the registration is complete.

(9, 10) The AMF contacts the PCF chosen at (7) and registers the UE If the PCF decides that the UE policies the UE has last seen are out of date, it sends an update.

4.3.4 PDU Session Establishment Procedure

PDU session establishment, illustrated in Figure 4.3, is invoked when the UE wants to establish a data session (PDU) with a DN it is subscribed to.

This section shows the establishment of a tunnel in the data plane. Of importance is how the UPF subscribes to changes in the usage reports located in the data plane, and to the spending limit, that can be modified by all PCFs serving this subscriber.

(1) The UE requests a PDU session to a data network identified over a specific slice. The UE also chooses a PDUid that will identify this PDU session later.

(2) The AMF selects an SMF to handle this session. This selection depends on the target DN and the slice chosen. The SMF will not change for the lifetime of the session.

(3) The SMF received a request to set up a data plane connection from the DN to the UE Location. If the DN is a LADN the AMF also informs the SMF whether the UE is close enough to connect. The SMF will create a Session Context to store the

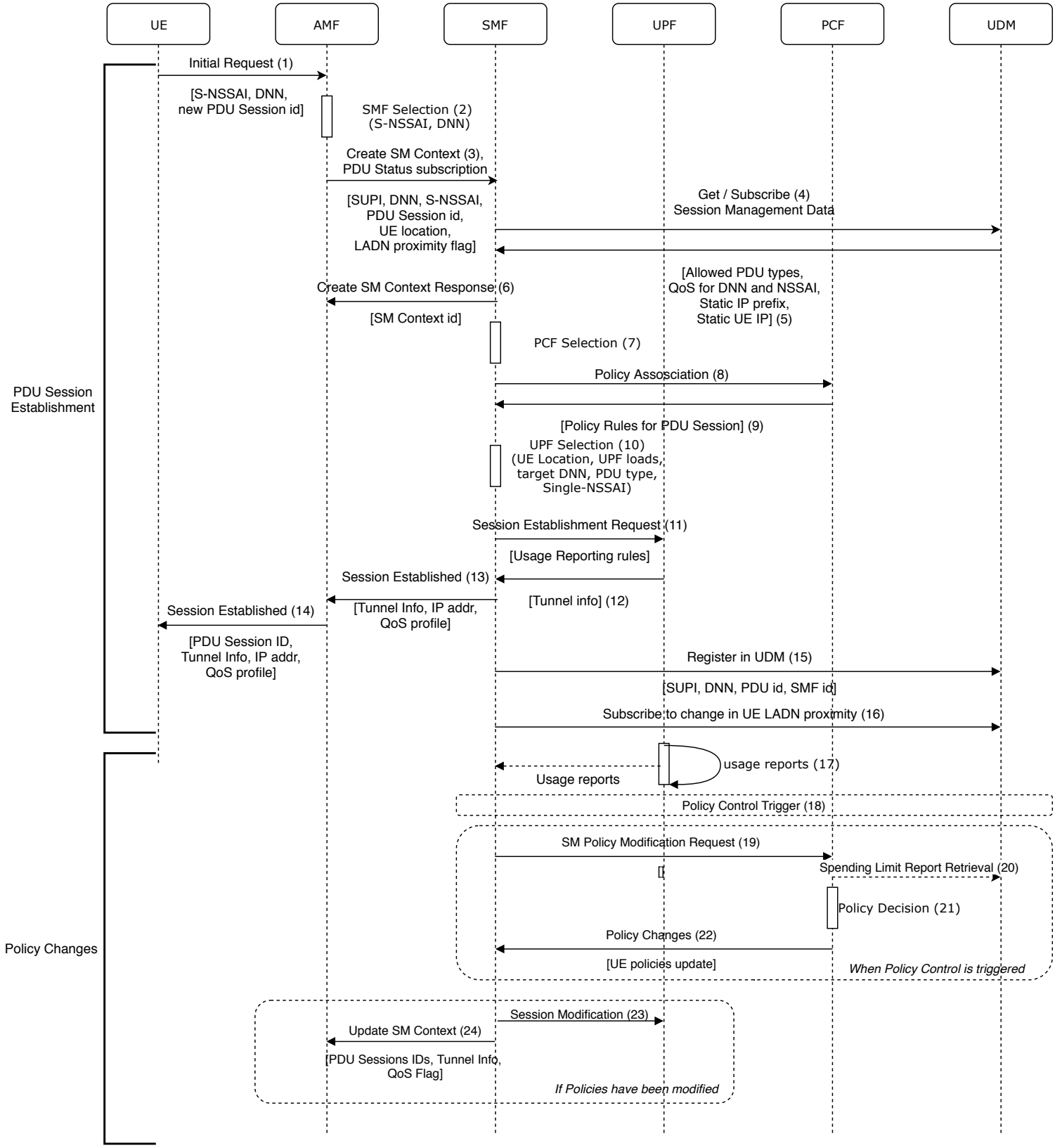

Fig. 4.3.: PDU Session Establishment procedure[2]

information regarding this session and the AMF subscribes to modifications to it during this request.

(4, 5) The SMF requests and subscribes to a portion of the UE Context that is session management specific (not kept by the AMF from the UDM). This information is the quality of service that is required when accessing the target data network over this slice, the PDU types that are allowed for this UE and, when it applies, whether this specific UE has a static IP address (or a static IP prefix) for this data network.

(6) The SMF returns its internal context id for the data session context to the AMF

(7,8,9) The SMF selects a PCF that will be responsible for enforcing the policy rules that apply to this PDU Session (See 18-24 for future policy updates).

(10) The SMF selects one or several UPFs . The role of an UPF is to provide connectivity by forming a chain in the plane. The first UPF to be chosen will not change during the lifetime of the session (like the SMF). It will connect directly to the data network and act as a session anchor (see PSA): when an IP address is chosen, it will point to this PSA. More additional UPFs can be chosen, these Intermediate UPF (I-UPF) will form a tunnel between the PSA and the UE. The UPF are chosen according to their location, the location of the UE and their current workload.

(11,12) The SMF request the setup of a data tunnel from the UPF and gives them instructions on how often they must report the session's data usage. The UPFs return tunnel information that will allow the UE to send data along this newly allocated tunnel.

(13,14) The UE and the AMF receive information regarding the tunnel address, the QoS profile of the newly established connection and the IP address of the connection in the target DN.

(15) The SMF registers itself in the Subscriber Data Management (SDM) data of the UDM along with the PDN name, the subscriber id and the session id.

(16) The SMF subscribes so that if the UE should be registered by the AMF in another location it will be notified.

After the establishment of the session

This is the start of the second procedure request. It shows how usage data is updates through regular usage reports, and how the PCF responds to updates in the usage and policy data it has subscribed to.

(17) According to the reporting rules sent during step 11, every UPF sends regular reports to the SMF

(18) Three mechanisms can trigger a policy control: (i) The SMF has received enough usage reports to trigger a check or it has received a notification from the UDM for a location change in the UE (ii) An unspecified internal event, in the PCF triggers the policy control (*e.g.* this could be a timeout). (iii) The UDM sends a notification to the SMF that triggers a policy control.

(19) If the trigger originates from the SMF a request is sent to the PCF for an update.

(20, 21) The PCF retrieves a spending limit from the UDM and makes a policy decision accordingly.

(22) The PCF informs the SMF of any changes to the policy.

(23,24) If change must be made to the session to comply with the policy, the SMF makes the necessary changes to the data plane and informs the AMF of any notifications.

4.3.5 Handover Procedure

The handover procedure, illustrated in Figure 4.4, is invoked when an UE is already connected and authenticated in the network. The UE connects to a new RAN. This RAN is the responsibility of a new AMF. To simplify the UE moves closer to a new antenna, it will connect to and register with this new antenna.

This section describes the transfer of data and responsibilities between AMF_{source} and AMF_{dest}. All the data held by AMF_{source} is sent over to AMF_{dest} who populates its UE context with it. The AUSF re-authenticates the UE as needed.

Once the registration to AMF_{dest} has completed, the two AMF modify the UE context information that is stored in the UDM by registering a new AMF id. The currently serving AMF must be kept up to date on the subscriber data. This is why AMF_{dest} subscribes to be notified of all changes and AMF_{source} must unsubscribe from the modifications it is no longer interested in. It also deletes its copy UE-context once the transfer is done.

(a) The UE requests to be registered with the AMF_{dest} in the same fashion as during initial registration. Because it was previously connected to the system, it still holds a local copy of some of the data in its UE Context.

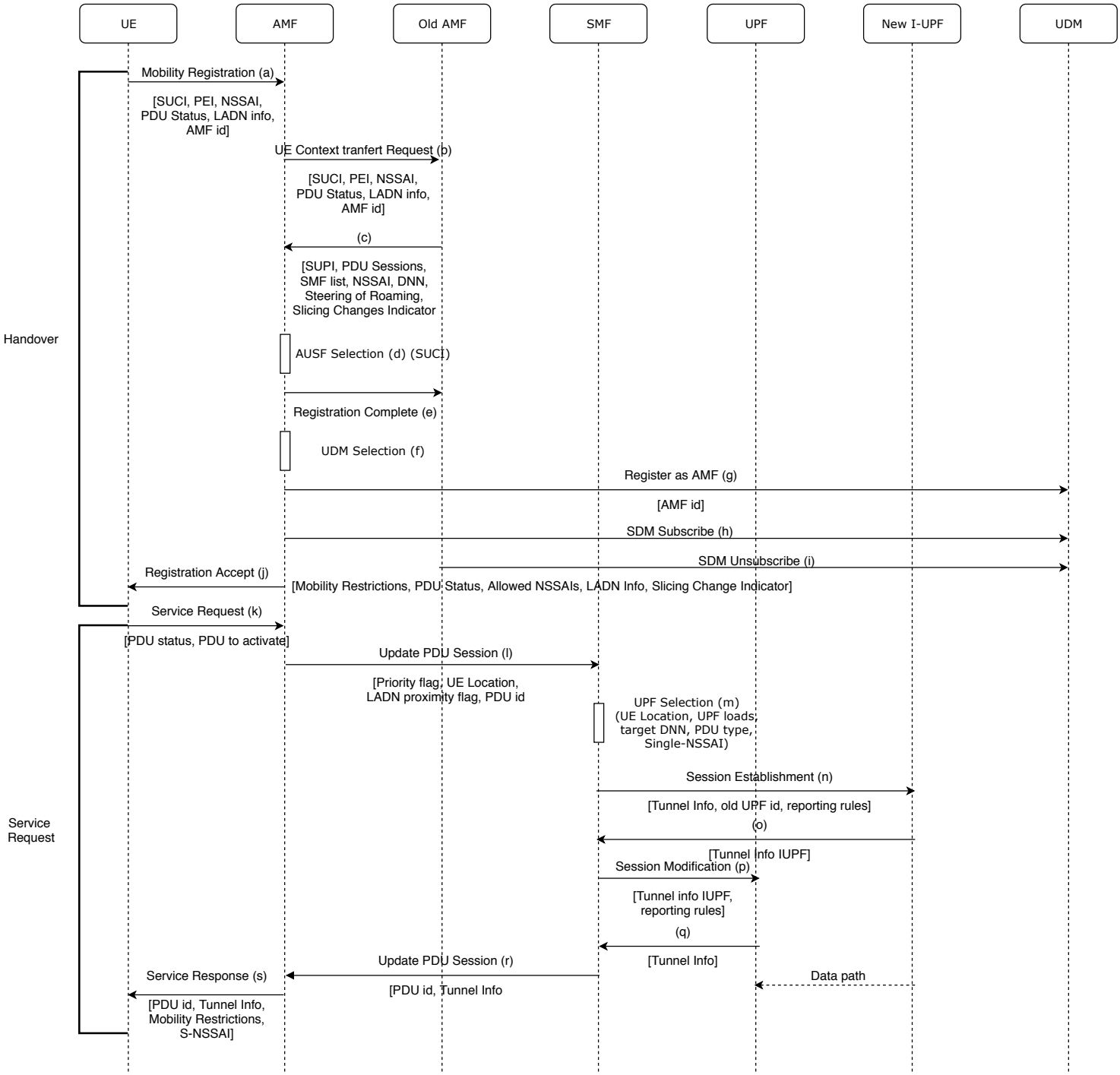

Fig. 4.4.: Handover procedure[2]

(b) The AMF_{dest} uses this information to identify AMF_{source} and sends a request for the UE-context.

(c) AMF_{source} sends what data it keeps to AMF_{dest}. AMF_{source} now knows that an UE it was responsible for has moved away.

(d) AMF_{dest} uses the Subscription Concealed Identifier (SUCI) to identify the AUSF responsible for the UE If necessary, the UE may be re-authenticated by the AUSF at this point.

(e) AMF_{dest} confirms to AMF_{source} that the handover registration is completed. At this point the AMF_{source} is no longer responsible for the UE

(f,g,h) AMF_{dest} identifies the UDM where the UE-context resides. It modifies the value of the responsible AMF within the UDM to itself and registers for future updates.

(i) AMF_{source} unregisters from future updates. AMF_{source} can now delete its copy of the UE-context.

(j) The handover registration is complete. AMF_{dest} sends the UE all the information it needs, as it did during the first registration request.

The second part of the handover procedure occurs when the UE requests access to any previously configured data session. In this part of the procedure, the UPFs that are used to connect the UE to a target DN are rearranged to reach its new location. This modification occurs as needed when the UE tries to use the connection, not automatically when during the first part of the handover.

(k) The UE sends a request with the identifier of an already existing PDU Session.

(l) The AMF contacts the SMF that is associated with the PDU Session in the UE Context.

(m) According to the new UE location, the SMF selects one or several UPFs that will bring connectivity to the UE. LADN are limited to a specific geographical zone. If the proximity flag is no longer true, the SMF will simply terminate the session.

(n,o) One or multiple new Intermediate UPF (I-UPF) connect to UPF that are already set up. They create a new data tunnel in the data plane.

(p,q) Some of the old UPFs are no longer needed and are freed. The others are given tunneling information to connect to the new I-UPF. Even though all other may be removed from the PDU Session, the PSA is not freed for the lifetime of the session. This is because it is directly connected to the DN that the UE is trying to reach.

(r,s) The SMF returns updated tunnel information that may be used by the UE to send and receive data over the PDU Session. The session is now set up correctly to reach the UE's new location.

Now that we can see all these steps, we can do our analysis of the 5G specification data.

4.4 Invariants and anomalies in 5G

Consistency mechanisms enforce invariants on the order of operations in the history of a data store. When an application access data through a consistent data store, it can avoid seeing anomalies in the data being returned. The order of operations determine what kind of anomalies are avoided.

To keep 5G available during network partitions means that we cannot afford the synchronization costs of a strongly consistent system, where all anomalies would be avoided. Since cannot specify a set of consistency semantics for the globality of the 5G data, we propose instead to build back from a correct behavior of the system. Try to detect points where an anomaly occurring would make the system behave incorrectly. And eliminate this specific anomaly in this part of the system. We will state application-level invariants and show what consistency mechanisms would enforce them.

4.4.1 Methodology

In Section 4.2.1 we listed the patterns of use, we expected to see in the 5G procedures. We propose to link variables to these patterns and to the procedures.

Tables 4.1, 4.2, 4.3, 4.4, and 4.5 each present a subset of the variables we extracted from the procedures. Variables with similar patterns are presented together. And for each individual variable we show which VNF uses it. 5G is built around a publish/subscribe pattern to share data, this is why we denote that a VNF is liable to modify and publish data with P for publish. A VNF that is liable to subscribe to a variable is noted C for consumers.

Most data is stored in the UDM, and it subscribes to all changes, except for ephemeral data plane data that is not intended to be propagated outside of the data plane. The UDM is a publisher for all variables unless specified otherwise.

Variable	UE	AMF	SMF
SUPI		C	
Static UE IP	C		C
Security Context	C	C	
Steering of Roaming	C		
Mobility Restrictions	C	C	
Slicing Change Indicator	C	C	
Allowed PDU Types			C

Tab. 4.1.: Constant Data or Subject to Manual Change (C: Consumer)

Ultimately the UDM is the publisher of all variables. This is why when a function is not primarily responsible for modifying a variable, no VNF is marked as P.

4.4.2 Local vs global invariants

There is a relationship between the invariants we want to enforce and the strength of an appropriate consistency model. We will not be able to enforce a global invariant without reasoning about the state of the system. And all consistency models that can enforce a global state are unavailable during partition.

The correctness of the system on a global scale must be enforced through local invariants. This is why the locality of the data is a key part of the consistency semantics we can choose for the procedures.

We show an example of whittling down a global invariant down to a local invariant that is sufficient for the correctness of the system in Section 4.4.4.

4.4.3 Infrequently changing data

The data presented in table 4.1 is either not liable to change, or may be modified manually from outside the parts of the system our analysis of the procedures reached.

We do not specify when these variables will change. However, once modified, the UDM should publish the new version. And from the moment when a read operation from a VNF returns the new value, all future reads must return the same value.

Variable	UE	AMF	SMF
AMF Identifier		P	
UE Location		P	C
LADN Proximity Flag		P	C
SDM Subscribed		P	

Tab. 4.2.: Registration-related data (C: Consumer, P: Producer)

To state this requirement as an invariant: *The network should not read an old version of a value if it has already read a newer version.*

This can be enforced through Monotonic Reads (see Section 3.1.6).

This first invariant may be enforced by a single-object consistency model. However, another element requires a multi-object model: the Slicing Change Indicator is a version number and serves to alert the VNFs when the state of slice-related information has changed.

An anomaly that could occur with a single-object consistency model is that updates to the Slicing Change Indicator and to the slicing-related information would become visible independently. In that case, the date could seem unchanged when read by a process, because the updated value is not yet visible.

When the change indicator is updated, the updated data should be visible for future reads.

Under Monotonic Writes (see Section 3.1.6) this could be avoided if the session that modifies the Slicing Change Indicator first modifies the related data.

Another option would be to update the data and the indicator atomically. Bailis et al. [8] propose Highly Available Transactions (HAT) under weak consistency models. These transactions would make the changes to Slicing data visible atomically.

4.4.4 Registration data

Table 4.2 shows data that is produced by the AMF during the *first registration* procedure. There is only one writer for this data: the currently serving AMF.

However the UE is a *mobile device*: the currently serving AMF is going to change, and its connection to the network may not always be stable during this change. Let us state a problematic scenario that could lead to a data race. A rapidly moving UE could invoke the handover procedure at two different locations, as it passes by. Two

AMFs would then attempt to register as the serving AMF. This would create a data race on the AMF identifier.

The result could be that multiple AMFs believe that they are responsible for the UE-context, or that no AMF is responsible. If no AMF believes itself responsible, it would then remove its copy of the UE-context and terminate any currently running PDU session.[5]

To avoid this situation, the system could enforce that: *For any UE that is connected, the network should know exactly one responsible AMF.*

However this statement applies globally. There is no way of enforcing it without relying on a strong consistency model. Our goal of remaining available during network partition would not be met. Instead we must find a local invariant that would be sufficient to maintain correctness. Here we want to avoid losing the current UE-context, and we also want to break a tie between two AMFs.

For any UE that is connected, at least one AMF knows that it is responsible for this UE and the UE knows this AMF's identifier.

A transactional operation could be used to update the value in AMF_{source}, AMF_{dest} and the UE at the same time. But this transaction would have to take place during the handover procedure. It would start with operation a and finish with operation j, when the AMF identifier is replaced in the UE. The UDM is also involved during step g,h, and i. It may have to be part of this transaction.

Such a transaction would be hard to set up in practice. The only actor that will always take part in this operation is the UE, and it is the least reliable one. By relying one the AMF identifier propagated by the UDM, the AMF_{dest} could read a stale value and request a copy of the UE-context from an AMF that has already deleted its copy.

To avoid having to rebuild the UE-context, there should be a continuous chain of serving AMFs. With the responsibility being passed from older AMF to newer AMF. It is not necessary for all actors in the system to reach a consensus on which AMF is responsible for any UE. There may be multiple AMFs holding a copy of the UE-context. It is sufficient instead that the AMF known by the UE be one that holds a copy that it can send to AMF_{dest} when the UE completes the registration process.

[5] Rebuilding the UE-context can take some time but is not the main cost of this untimely termination. The data plane can hold and redirect undelivered PDU, giving the illusion of uninterrupted connectivity to the subscriber.

Variable	UE	AMF	SMF
IP addr	C		P
S-NSSAI	P	C	C
DNN	P	C	C
PDU Session id	P	C	C
SMF id		P	
SM Context ID		C	P

Tab. 4.3.: Setup once at session request (C: Consumer, P: Producer)

By completing the registration, the UE breaks any tie between competing AMFs. Any AMF that holds a copy of the UE-context is also subscribed to updates on this data. A notification can be propagated by the UDM and any other AMF may then delete their copy.

When the UE connects to the network again, it can break any tie between competing AMFs. And as long as at least one AMF believes that it is responsible for the UE, a copy of the UE-context is retained.

4.4.5 Constant data for the lifetime of a PDU session

The data presented in table 4.3 is used during the setup of a new PDU Session. The values of all these variables are either obtained from the UDM during registration of the UE, or they can be generated by the UE (PDU Session id) or the SMF (SM Context ID). Here our requirement is that the data we obtained from the UDM was consistent. That the DN chosen is available on the slice designated with the Single NSSAI (S-NSSAI).

Afterwards, none of these values will change for the lifetime of the session. The only requirement for this data is that it be kept from all modifications until the SMF terminates the PDU session.

4.4.6 Ephemeral data for the lifetime of PDU Session

The data in table 4.4 is the only data that is never replicated in the UDM. Indeed these are also the only variables that are not intended to leave the data plane (with the sole exception of the SMF, who must still manage the UPFs in the data plane).

Variable	UE	SMF	UPF
Tunnel Information	C		P
UPF loads		C	P
SM Context Contents		P	
Usage Reporting Rules		P	C

Tab. 4.4.: Ephemeral Data Plane Data (C: Consumer, P: Producer)

Tunnel information for instance will be used by the UE and the UPF to communicate with each other. It does not make sense to replicate it further in the control plane.

The value of the UPF loads, however, affect the system in a wider radius. They are sent to the SMF so that it can make routing decisions for existing or future PDU Sessions. By finding an UPF that can support a new connection or by rebalancing the load when network traffic shifts. This routing information is specific to the physical neighborhood of the SMF, we can ensure that the view it has of the loads if free from anomalies by specifying that this part of the network should be is a HAC zone (see Section 3.2.2).

Another value that is specific to the UE's subscriber is that it affects the spending limit for this user. And a user may have several PDU session opened at any one time. For the spending limit to be accurate, it must respect the following:

The value of the spending limit will eventually reflect every local operation.

We do not know how distributed all the SMFs associated with a client may be, so we cannot ensure that this is true through a HAC zone. However, a spending limit is ultimately a grow only value. With a consistency model that requires strong convergence (CC+ or SEC) and a counter based on a CRDT, we can be sure that once updates from all SMFs have been made visible by the UDM, the Spending Limit will hold an accurate value.

4.4.7 Data modified when a PDU Session is used

Table 4.5 presents data that is affected by the data plane and that affects the data plane in turn. An interesting part of this data is the Spending Limit loop.

First, the PCF setups a trigger when the session is established. The subscriber uses the data session, and the SMF collects usage reports. When triggered (either from the trigger established here, from the SMF, or from a trigger propagated through

Variable	UE	AMF	SMF	PCF
QoS for DN and S-NSSAI	C		C	P
UE Policies	C		C	P
PDU Policies			C	P
Policy Trigger			C	C
Spending Limit				C
PDU Status	C	C	P	

Tab. 4.5.: Change when network in use (C: Consumer, P: Producer)

the UDM) the PCF collects the usage report from the SMF, updates the limit and sets the trigger again.

When enough network resources have been spent by the subscriber, and the limit grows higher, several things can happen. The triggers may occur more often, to ensure that the limit is not breached. The QoS can be updated, so that fewer resources can be spent between two triggers, or because the subscriber is now limited to a lower QoS. Or the session can be terminated.

Here the invariant is obvious for the application: *A limit that has been set should not be exceeded.*

The problem here, compared with Section 4.4.6, is that convergence is not sufficient to enforce this invariant. The value should still converge so that the limit reflects the actual usage of the network. Additionally, once the value has converged, it should not reach a value greater than the spending limit.

All PCFs associated with the subscriber should know the accumulated amount of network resources that have been consumed by the subscriber. And the different replicas should not diverge far enough that merging these amounts together produces a value beyond the Spending Limit.

For a subscriber using a single UE, a HAC zone could be sufficient for the PCFs to synchronize as the limit gets closer, and to launch a trigger on all PDU session when it is reached.

However, if a single subscriber has access to multiple UEs, or in the case of a Spending Limit associated with a slice, solving this can be more challenging. There are tradeoffs between the availability of the system and fidelity to the spending limit. Section 5 is dedicated to the methodology of assessing the performance of different limiter strategies.

5 Better Rate Limiting of Resource Consumption in a Distributed Network

In Section 4 we studied three mobile network procedures to present the lifecycle of data in the 5G network, and to highlight where consistency guarantees are necessary to preserve correctness. We proposed a set of invariants that are sufficient to preserve the consistency of data for these three procedures, as well as mechanisms that do not rely on strong consistency guarantees. To preserve the availability guarantees of the network.

In this section, we consider a network with distributed slices and explore further the final invariant presented in Section 4.4: *a limit that has been set should not be exceeded.*

Our first contribution begins with a description of why the enforcement of this invariant is more challenging in a network with slicing. To reflect on potential solutions to enforce this limit globally, we propose the limiter abstraction, a network component dedicated to preventing overuse of a slice's resources.

Our second contribution is a set of metrics that should be measured in order to evaluate the fitness of any potential solution. Limiters are to be deployed in the context of mobile networks, with potentially strict availability or responsiveness requirements. Enforcing our invariant is not always the priority for 5G designers and according to the use case, the tradeoffs of fidelity must be measured against the two other properties that can affect QoS.

Finally, we present rate limiting approaches used previously and consider the fitness of their approach before explaining the need for a solution that is built upon distributed data.

5.1 Limiting the rate of resource consumption in a distributed system

In this section we show why the enforcement of limits is more challenging in a network with slices.

Network Slicing[1] is intended for industry use cases and industrial clients. Where with regular subscribers, we can proceed with the hypothesis of them using a few UE devices placed within a relatively small geographical zone. Industrial clients are expected to require the deployment of large-scale slices, possibly over the whole coverage of the Telecom Operator; these slices will offer connectivity to fleets of devices[2].

In Section 4.4 we studied invariants that, either remained correct while replicas diverge, or relied on strong consistency in a localized geographical zone. But these are irrelevant here, because

- A slice can be deployed over a zone that is too large to enforce a stronger consistency model.
- Divergence between replicas can violate the limit.

A slice is configured to offer specific guarantees and SLA. These guarantees are implemented thanks to specific hardware, scheduling rules, routing algorithms, prioritized packets and other network resources.

In our model, a slice is setup on a metered connection. A set amount of resources is available for each slice, which once used up, turns off the slice. The PCF acts as the limiter interposed between the client and the slice. It regulates the global limit that applies to all devices connected to the slice.

An LTE/4G network already has the capability of enforcing a Spending Limit on a subscriber. The multi-device and distributed nature of a slice make resource limiting significantly more challenging.

A limiter should not only limit the total amount of resources consumed by the slice, but also the rate at which resources are consumed. For this purpose, we propose to divide time into discrete *epochs*, and to meter the amount of resources used in each epoch separately. The amount of resources available is reset at the beginning of the next epoch.

[1] Presented in detail in Section 2.2.
[2] See Section 2.3 for example use cases.

5.1.1 The Limiter Abstraction

We present the Limiter, an abstraction implemented in the PCF. The PCF allows requests up to a prescribed limit. Underneath this abstraction, we study different consistency mechanisms and limiter algorithms, each designed with a specific consistency goal and different synchronization needs. The properties that are of interest in each design are:

- *Fidelity*: How close is actual use to the prescribed limit?
- *Responsiveness*: What extra latency is introduced in the system by the presence of the Limiter?
- *Resilience*: How well does the Limiter function during a network failure?

5.1.2 Previous Rate Limiting Mechanisms

In this section, we study previous limiter mechanisms, in particular low-level packet flow control mechanisms used in networks [33]. This study demonstrates why these previous solutions are not appropriate for enforcing a global limit.

These solutions control the number of packets allowed to flow in the network, they can also moderate packet bursts and sustained rate, according to their configuration.

Token Buckets

A Token Bucket [32] limits the packets that can pass through it. A *bucket* stores *tokens*. A packet may pass through the bucket only if a token can be spent from the bucket. Otherwise, either the packet is kept until the token bucket is replenished, or it is discarded.

There are two ways to refill the token bucket. Either another network component sends new tokens to the bucket, or the limiter creates new tokens according to a time-based policy. For instance, the limiter can be configured to refill its bucket at the end of every hour.

Leaky Token Buckets

Where Token Buckets limit the number of packets, a Leaky Token Bucket limits their rate. A Leaky Token Bucket empties through a hole in its bottom. Tokens fall through the hole at a steady rate, leaving free space in the bucket. A packet is allowed to pass through only if the bucket is not full. The properties of a Leaky Token Bucket are the following

- The size of the hole is the sustainable rate of the connection
- The size of the bucket is the largest size allowed for a burst of packets

These parameters make the Leaky Token Bucket versatile. It offers fine-grained control of the network traffic. Its configuration is complex. To configure it properly, the network's sustainable and burst rates must be known during setup.

A bucket limiter is self-contained. Once the bucket is full, it remains available, even during network partition. In a large-scale network, where partitions will occur, this makes buckets a resilient approach. However, as we will see in the rest of this thesis, isolated buckets cannot properly enforce a slice-wide resource limit.

5.1.3 Fitness of token limiters

In this section we state why the flow control mechanisms presented above are insufficient.

Consider two clients, trying to use a metered network resource, from two locations, such that their communication paths are disjoint. The limiters on the two paths must share the amount of resources spent so far if the limit is shared.

To illustrate, consider two Token Buckets, each initialized with the total amount allowed. In this case the *effective limit* is twice the *configured limit*. Because each of the two clients could spend the totality of their local bucket. Consider instead the same configuration, but with two limiters assigned half of the allowed amount each. In this case, the sum is equal to our *configured limit*, but if only one of the clients is active, it will be limited to half the target amount, even though the other one spent nothing. Thus in the worst case, the *enforced limit* is half the *configured limit*. In conclusion, a limiter made of two independent buckets does not work for slices.

Note, however, that buckets remain available even during network partitions, and that a token bucket respond to its requests locally, without a need to synchronize with

other replicas. This ensures that no latency overhead is introduced by communication delays, and that the responsiveness of the slice is not affected by the addition of a limiter.

Therefore we conclude that fidelity requires that individual limiters communicate, in order to enforce some consistency guarantees. A network of local limiters, or *replicas*, work together as an abstract, distributed limiter.

However, introducing communication between the replicas can impact responsiveness and availability for the slice. The tradeoffs that come with each protocol should be measured in accordance with the use case's requirements.

5.2 Fitness of a distributed limiter

To properly enforce spending limits for slices, a limiter should have the following properties:

- A network of replicas acts as a single, global limiter, in order to enforce a global limit.
- During normal use, the replica adds little latency.
- During a network partition, the replica remains available, and responsive. However, being available should not prevent the limiter from enforcing the global limit.

Figure 5.1 show how a limiter interposed between clients and a slice. When a client submits a request, it goes through a replica. This replica decides if fulfilling this request will make the slice consume more resources than allowed. A client is not aware of the limiter's decomposition into replicas. The client submits a request to a replica, which accepts or reject it. To make its decision, a replica may rely on local state, or communicate with other replicas. It is only by communicating that individual replicas behave like a global limiter and enforce a global invariant.

As mentioned in Section 3.2 the CAP Theorem prevents a limiter from achieving all the qualities of fidelity, responsiveness and resilience. The fastest limiters will likely not have the highest fidelity. A limiter that is more resilient to network partition will not stay correct, because its replicas kept working despite being unable to contact other replicas.

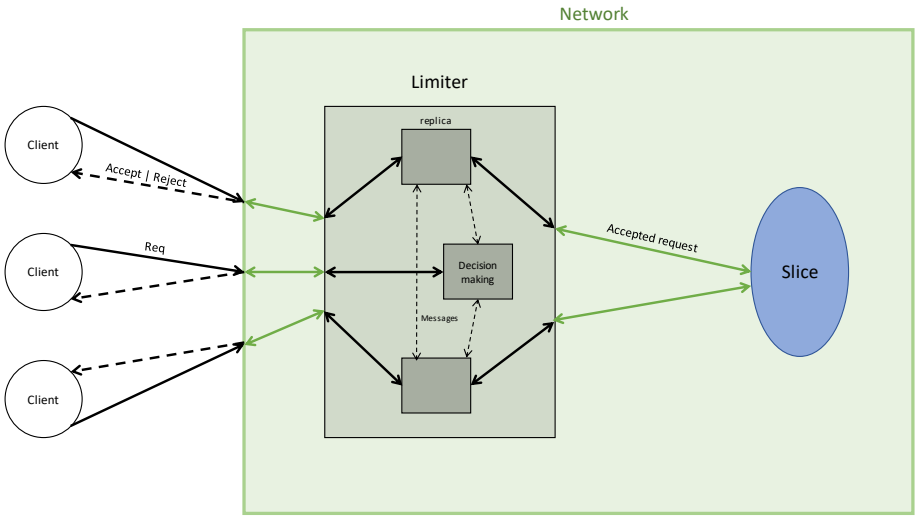

Fig. 5.1.: Limiters filtering user requests to enforce a global limit.

5G is designed to adapt to the future requirements of its users, even though they may yet be unknown. Slices are destined to be used by industries, and their future real-world requirements. Solutions for enforcing limits as strongly as possible exist in the form of Strong Consistency mechanisms, but these may not be compatible with other connectivity requirements of the slices.

A study of the consistency requirements to enforce distributed spending limits should not aim to provide a perfect solution of either fidelity, responsiveness or resilience. Rather it should propose a portfolio of solutions and present their tradeoffs. This way, system designers for future networks will be able to select a solution according to the use case each slice aims to serve.

6 Distributed Limiter Algorithms

We have shown in Section 5.1.1 that isolated replicas cannot enforce a global limit in 5G. In this section we present distributed protocols for limiters that address this problem.

We describe four protocols distributed at interesting points of the consistency spectrum as well as the network model we choose for their design. The local replicas communicate with one another under a replication protocol, acting as a global limiter.

6.1 Design decisions

In this section we present our assumptions regarding the 5G deployment for which we designed the limiters.

The mobile network is composed of three kinds of elements: antennas, edge server and cloud servers. Antennas are widely distributed. Each antenna is connected to a local edge server. Conversely an edge server is connected to an antenna, and to a cloud server. A cloud server is connected to multiple edge servers, and to all other cloud servers. A local limiter replica is co-located with each server, so that one replica is placed in each edge server and each cloud server.

A user sends a request to the network through the nearest antenna. The edge server that is connected to this antenna receives this request, and to ensure that the global limit is respected, this request if filtered through the local replica.

The global spending limit is an integer value. A request consumes an integer amount of resources in an all-or-nothing fashion. The local replica decides whether the current request will exceed the spending limit, and it accepts or reject it in consequence. To make this decision, the replica can:

- Act as a proxy to another authoritative replica and pass the request along.
- Synchronize with other replicas to make a decision.

- Rely on its local data and make a decision locally.

Time is divided into discrete *epochs* of some duration. The limiter enforces its limit per epoch. Epochs are independent, and they do not overlap. At the start of a new epoch, the limit is reset to its initial state and can be spent once again. The unused credit from a previous epoch does not carry over to the next epoch.

The rest of this chapter is dedicated to a description of the design of each limiter and its guarantees. A more precise description of each limiter is available in Appendix B.

6.2 Notation

For the remainder of this chapter, we note *cap* the value that represents the global spending limit to be enforced per epoch. The network is composed of N servers. Each server has its own local replica. The set P of N replicas is noted $P_0, \ldots, P_i$

We characterize each limiter algorithm with the following metrics:

- *resilience* to network partitions depends on whether a replica responds to queries when isolated.
- *responsiveness* of a limiter measures the average response time of client requests.
- *fidelity* is the ratio between the number of requests it allows and the number of requests it should have allowed. To compute the average fidelity during n epochs, we note the configured limit cap, and the effective use during epoch i: use_i.

$$Fidelity_{avg} = \frac{\sum_{i=0}^{n} use_i}{n * cap}$$

  A fidelity of 1 is optimal, it means that the resource has not been overspent and that all the requests that should have been approved are approved. Below 1, the limiters has spuriously rejected requests that should have been approved. Over 1, the limiter has not successfully enforced the invariant, and the resources have been overspent.

6.3 Central Leader Limiter (CL)

Our first design is the Central Leader Limiter. As the 5G specification makes strong consistency requirements, our first design is a strong consistency limiter, strong consistency guarantees that the spending limit will be respected.

The Central Leader Limiter consists of an authoritative replica, every other replica serves as a proxy that forwards its requests to the authoritative replica and relays back the latter's response.

A replica P_{leader} is elected, either at initialization or at some other point if the leader is dynamic. P_{leader} holds a variable *credit*, initially set to cap. When a replica P_i receives a request, it forwards it to P_{leader}. P_{leader} accepts this request if *credit* is greater than zero, and decrements *credit* by one. Otherwise P_{leader} rejects the request.

With CL, all operations are centralized at the authoritative replica, which totally orders them. Strong consistency guarantees this total order, and the limiter provides high fidelity, at the cost of availability. If P_{leader} were to become unavailable, no request would be granted and the slice would become unavailable.

We expect that the fidelity of this limiter will be equal to 1. Its response time will depend on the maximum Round Trip Time (RTT) between an edge server and the authoritative replica. Because of its centralized nature, this limiter will not be resilient.

6.4 Strongly Eventually Consistent Limiter (SEC)

To provide a high level of resilience, we propose the SEC limiter, with weak consistency guarantees. The aim of this design is to provide the strongest resilience guarantees, at the expense of fidelity to the global limit.

The SEC Limiter holds less guarantees over the consistency of its state than the Central Leader Limiter. Each replica can modify its state independently, and the guarantees reside in the state of all replicas eventually converging to the same value

through a merge operation. However, unless requests are very infrequent or very localized, it is unlikely that the limit is enforced[1].

A replica of the SEC Limiter contains *spent*, a vector of N elements, initially zero. Entry spent[i] is the replica's estimation of the resources spent by replica i. The total amount spent is the sum of all elements in the array $\sum_{i=1}^{N} spent_i$. The remaining credit is $\sum_{i=1}^{N} spent_i - cap$

When limiter P_i receives a request from a client, it computes the current value of the spending limit. To do so, it subtracts the sum of all elements in *spent* to *cap*. If the result is positive, the request is granted and the limiter increments element i of *spent* by one. Otherwise the request is rejected.

Each time a local limiter grants a request, it sends an update message to all the other replicas. This update message contains a copy of the *spent* vector.

When a limiter receives an update message containing the sender's spent vector, it merges the vector received into its local *spent* vector. This operation computes the least upper bound of the two vectors, which is the maximum of the local value and update's value. When the current epoch has ended, all elements of spent are reset to zero. If an update has been sent during a previous epoch, it is discarded upon arrival, because only operations from the current epoch can affect the spending limit.

We expect that the fidelity of this limiter will not be enforced, and that the result will be above 1. However the response time should be very short, with no overhead introduced by the limiter.

6.5 Bounded Counter Based Limiter (BCL)

This limiter is intended to provide stronger guarantees than the SEC Limiter while still remaining as resilient as possible. This is why it relies on as little communication between replicas as possible.

The BCL limiter is based on the design of the CRDT Bounded Counter [9], as well as the Token Buckets presented in Section 5.1.1. Here the limit is enforced through the use of tokens, tokens are exchanged between replicas and cannot be duplicated. The number of tokens is fixed and does not change after initialization. One token

[1] Infrequent requests allow time for updates to be propagated to all replicas, and localized requests centralize use of the limiter.

must be spent to authorize each request, and it may only be spent once during an epoch. At the end of an epoch, all tokens are renewed and can be spent again.

At initialization, an arbitrary replica is created holding as many tokens as the global spending limit allow requests per epoch. Every other replica is created without any token. An unspent token may either be spent locally by a limiter, or it may be transferred to another limiter on demand. A spent token may not be transferred to another limiter.

Each limiter holds a two-dimensional array of tokens, N^2 elements which describes how tokens are exchanged between limiter replicas. It also holds an array *spent* of N elements which describes how many tokens each replica has already used. This vector *spent* is discarded at the start of a new *epoch*.

When a limiter receives a request, it spends a token if it holds any. Otherwise it requests a token transfer from another limiter. It chooses this limiter based on the knowledge of where tokens are currently placed and how many have already been spent. A limiter that accepts this transfer modifies *tokens* accordingly and sends a copy of *spent* and *tokens* with an update message. Upon reception of an update message, a merge operation is applied by keeping the maximum value of every element of *tokens* and *spent* between the local value and the copy stored in the update message. However, if the update originates from a previous *epoch*, *spent* is discarded. The effect of not discarding *tokens* at the start of the new epoch is that tokens are held in the replicas where they were last spent. This means that the response time of the limiters can be shortened when requests take place in the same location.

We expect that the value of fidelity will not exceed 1 The fidelity of the limiter is enforced by the unicity of tokens. The response time is harder to predict, it depends on the locality of requests. If a limiter holds a token, it does not need to contact another replica to obtain a token. However, if it holds no token, the response time will depend on the latency between the local replica and the replicas it contacts to obtain a token.

As for resilience, replicas of the BCL Limiter do not need to synchronize with a centralized replica. Even if the network is partitioned, as long as the replicas that can still communicate with each other hold some tokens they will still be able to function.

6.6 Predictive Presence Based Limiter (PPB)

Our objective with this limiter design was to design a limiter that relies on the nature of the network to avoid communication between replicas while still enforcing the global limit. We want to see if we can design a limiter with good fidelity around a simple heuristic.

The PPB limiter is based on our observation that in a 5G deployment, antennas can detect how many devices are currently connected locally. We also rely on the hypothesis that the total number of devices that can connect is known.

At the start of each epoch a limiter queries its antenna for the number of devices currently connected to it. It calculates the proportion of devices connected out of the total number of devices, and allocates itself a proportional portion of the global spending limit. When a limiter receives a request, it spends from its local share to approve or denies it.

This limiter only relies on its local state and local information to function, we expect that will have the same response time as the SEC Limiter and that it we be resilient to network partition. For fidelity, if each limiter only allocates itself a part of the total limit, the fidelity should be measured at or below 1. However to be sure that effective use matches the configure limit means that there should be a strong link between the locality of the device and requests. It means that every device currently connected needs to make a request locally, and that every limiter receives a request.

6.7 Discussion

We proposed a few scenarios in Section 2.3 where several slices had different requirements and different locality patterns. We now quickly discuss how the design of these limiters could each be paired with a use case.

For the Itinerant Slice, the BC Limiter seems appropriate as tokens could migrate in the same pattern as the users. The area over which the slice could potentially be deployed is large, but only the replicas that hold a token would be involved to limit use of the slice.

For the Industrial Slice, a CL Limiter could be chosen, with a leader chosen strategically to minimize latencies for most clients. If the different locations are too far apart and the response time too long, we could rely on the limited number of antennas and clients and use the PPB Limiter.

Finally the Critical Slice has very different design constraints. Even though potentially less spread out than the Industrial Slice, communications must be as fast as possible, and fidelity is *not* a priority. Instead, the SEC Limiter could prove to be perfectly suitable with its ability to reflect actual use once all replicas have converged, without incurring additional response time.

The next chapter focuses on an experimental evaluation of these algorithms. The behavior of the limiters during a simulation could validate these pairings.

7 Evaluation

Previous chapters focused on the qualitative aspects of solutions to enforce invariants in the 5G network. However, algorithms for the global enforcement of a spending limit come with unavoidable tradeoffs, which are hard to predict, and are highly dependent on the use case. They can affect the Quality of Experience (QoE) of the network by introducing delays or spuriously limit users if the effective limit turns out to be lower than the set limit.

We propose to measure these tradeoffs experimentally. Due to the lack of a large-scale 5G core network deployment, or even a simulator, we modify an existing simulator with the intent to make the simulation behave in the same way as the 5G network.

The contents of this chapter are presented as follows:

- Section 7.1 presents how a simulation scenario is run. To understand how our simulator behaves and what role the limiters play in the simulation. We present the settings that affect the behavior of the simulation.
- Section 7.3 presents the results of running limiters according to the protocols presented in Section 6.

Our simulator is named SliceCap [28], it is intended to evaluate the performance of limiters to cap a slice's access to resources. This discrete event simulator is built on top of EdgeCloudSim [29], itself an extension of the CloudSim [13] simulator.

7.1 Presentation of SliceCap

The simulation models a set of mobile devices and a two-dimensional world in which antennas are placed. Mobile device move according to their mobility model and make requests of the network at random intervals. These requests are how network resources are consumed.

Each mobile device is dedicated to an application. An application corresponds to a use case, it describes how often a device can be expected to make requests and how much bandwidth and resources each request will consume. For our simulation, we consider that each application consumes resources in its own slice.

Antennas are placed in the two-dimensional grid of our simulation. Each antenna is controlled by an edge server, and each edge server is connected to a cloud server. Servers communicate by sending messages, the communication latency between them is equal to the distance between their position on the grid. All cloud servers can communicate with each other, and with the edge servers they are connected to. Edge servers can only communicate with their cloud server. All edge-to-edge communication must pass through the cloud layer.

For the length of the simulation, time is divided into *epochs* of equal length. An epoch is the time during which a slice's spending limit has to be enforced. After each epoch, the spending limit is reset to its original value.

To enforce the spending limit, a local replica is placed in each edge and cloud server. Local replicas act as a global limiter by working as described in Section 6. The latency of messages exchanged between replicas is the same as latency between the two servers they are located in.

A mobile device makes a request randomly and sends it to the closest antenna. The antenna receives this request and filters it through its local replica.

7.1.1 Important variables

In this section, we list the parameters we vary in the simulation in this section, as well as the impact they are expected to have on the simulation:

First the parameters of the simulation itself are:

- The duration of the simulation.
- The number of devices that will be simulated.

Second we configure the topology of the two-dimensional grid in which the mobile devices move. These parameters are:

- The position of the antennas in the grid, and so the position of the edge server each is co-located with.

- The cloud server to which each edge server is connected.
- The attractiveness of the antennas: this affects the mobility of the mobile devices. They follow a random waypoint model where each antenna is a waypoint. The attractiveness of a waypoint affects the resulting mobility traces.

Each device is dedicated to an application, this application describes how devices are expected to make requests of the network. The parameters of the application are:

- The global spending limit for this application. Each application consumes resources in its own slice, and so each has its own spending limit.
- The length of an epoch: the duration during which a limit must be enforced. This is the length after which the global limit is reset and the application may once again consume resources.
- The interval between two requests: all devices make requests independently, and the interval between two requests is generated with a Poisson distribution. Lowering the means of this Poisson distribution increases the total number of requests during the simulation.
- The limiter algorithm used to enforce the global limit, as described in Section 6.
- A percentage of devices for each application can be made immobile for the duration of the simulation.

  The idea behind this parameter is that we wish to specify the percentage of requests that will be made in a location in which a request took place during the previous epoch. Making a percentage of devices immobile during the simulation is an approximation of this idea.

These parameters influence the way the simulation unfolds in several ways. We note two of them:

The contention on resources is affected by several settings. The global limit and the number of devices, because they describe how many devices will share a number of resources. Contention is also affected by the length of the epoch, because if the epoch is lengthened then resources are renewed less often, and so fewer resources will be available over the duration of the simulation. The Poisson mean also determines how many requests each device will make during the simulation.

Second we note the visibility of operations on the network is also affected by the length of the epoch and the size of the simulation area. If an epoch does not last as long as the RTT of the simulation, then for all messages exchanged between two replicas at the far ends of the grid, an update that describes their state will be out of date by the time it arrives at its destination.

7.1.2 Modification made to EgdeCloudSim

EdgeCloudSim[29] has served as the basis for SliceCap and it has been necessary to make modification to this simulator for this thesis. The modifications of notes are as follows:

- In EdgeCloudSim the cloud layer consists of a single datacenter. We added an additional topology of cloud servers through which all edge-to-edge communication pass, and in which a limiter replica may be placed. In 5G the core network is supposed to be deployed with multiple datacenters, and not with a centralized topology.
- We added limiters as a component of the network. Before running, a task must go through an additional processing step where it is submitted to a limiter. The design of the limiters corresponds to what we present in Section 6.
- We changed the mobility model of the simulation. Mobile devices can no longer teleport randomly from location to location. We now generate a mobility trace with the BonnMotion [7] software at the start of the simulation to describe the mobility of devices. We use a Random Waypoint model [12] for the devices that are not configured to be immobile during the simulation.

7.2 Topology of the simulation

In this section, we describe the topology of the simulation and the values we have chosen for the variables.

The simulation takes place in a 2D area of dimensions 1400, 800. Five cloud data centers are placed randomly in this area, and a total of 75 antennas are distributed around these data centers. These five data-centers play the role of cities that attract and retain mobile devices for a time. For each data center, five high attractiveness antennas are distributed randomly at a distance of 5 to 35. Ten low attractiveness

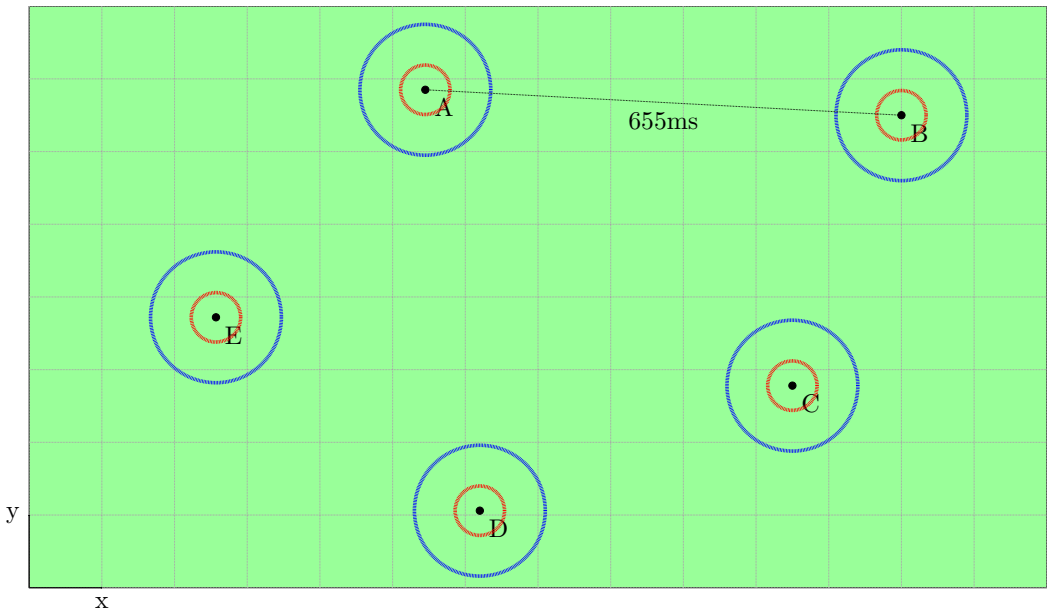

Fig. 7.1.: Datacenters placements in the simulation

antennas are distributed farther, at a distance from 45 to 90. Figure 7.1 illustrates this topology by showing the placement of the datacenters. The antennas of high attractiveness are placed within the red circle, the antennas of low attractiveness are placed within the blue circle.

The latency of messages exchanged between cities A and B is shown on Figure 7.1 for information. As described in Section 7.1, the latency between all messages is computed from the distance between them.

7.3 Measurements

7.3.1 Response time

To assess the response time of the limiters, we use the settings described in Table 7.1. We choose to simulate 75 antennas to make the world big enough to be representative of a 5G deployment. The epoch is shorter than the maximal RTTs to prevent all servers from communicating within one epoch.

Parameter	Value
Antennas	75
Epoch Length	500
Global spending limit	150
Mobile devices	200
Poisson mean	100
Immobile devices	0%

Tab. 7.1.: Simulation parameters

There are more devices than allowed requests during the spending limit to provoke contention, and the Poisson mean of the requests is short enough to be sure that requests will occur at several points during an epoch. Finally, no device is made immobile during this simulation.

Figure 7.2 plots the response time of each limiter algorithm when used in this simulation. This response time is the overhead added to the system by the presence of the limiter, *i.e.* how long it took for the replica to take its decision. The horizontal axis shows the response time of the limiter, while the vertical axis shows the cumulative distribution of this response time for the limiter. For instance, for the central leader no request took longer than 1400ms, and about 20% of requests took less than 500ms.

The PPB Limiter and the SEC Limiter have no measurable overhead in this simulation. The BC Limiter has no overhead for the majority of requests, but for a little percentage of the requests. When overheads occur, they are the most expensive, at up to 2300ms. The Central Leader shows a measurable overhead for all requests, with 60% taking between 1200ms and 1400ms.

We can conclude from these measures that the Central Leader Limiter has the highest overhead overall. That is explained by the fact that clients that are far away experience greater communication delays. Clients that are in proximity of this central replica get good service and a timely response. But as they get further away, response time gets larger.

The BC Limiter is the second in terms of overhead, because it can experience high response time when it needs to retrieve a token from another replica. However this overhead is offset by the ability of the replicas to reuse tokens. When one is available, a replica can rely on its local data to answer a request, and the response time gets shorter.

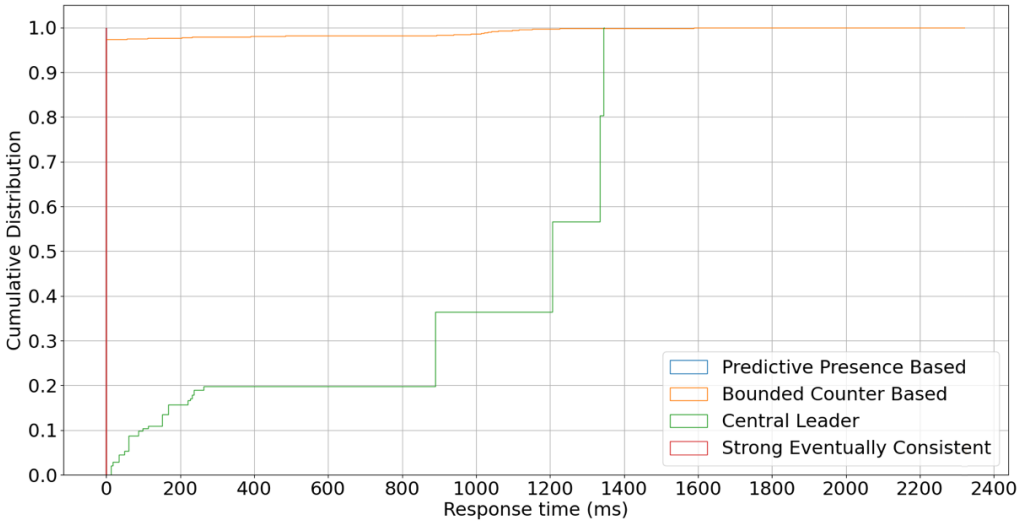

Fig. 7.2.: Response time of the limiters

The Strong Eventual Consistency (SEC) Limiter and the Predictive Limiter do not communicate with other replicas when they answer a request. Which is why their response time is not measurable.

However, this great response time is offset by these limiters fidelity to the global limit, as we show in the next section.

7.3.2 Fidelity to the global limit

Figure 7.3 presents results from the same simulation run as Figure 7.2. For each epoch of the simulation, we plot the ratio the fidelity of the limiter by dividing the number of requests that were approved by the configured limit. The $Fidelity_{avg}$, as defined in Section 6.2, is shown in Table 7.2 for every limiter.

The SEC Limiter is the only limiter to be above 1. On average, the SEC Limiter admitted 8.36 times more request than what should have been allowed by the global limit. The results of the SEC Limiter distort the rest of the plot, which is why we zoom on the other results in Figure 7.4.

The PPB Limiter reliably approves about 77% of the requests that should have been admitted according to the global limit. The BC Limiter sometimes approaches the ratio of 1, but overall it only admits 99% of the requests it should. The Central Leader Limiter behaves optimally, by perfectly enforcing the global limit.

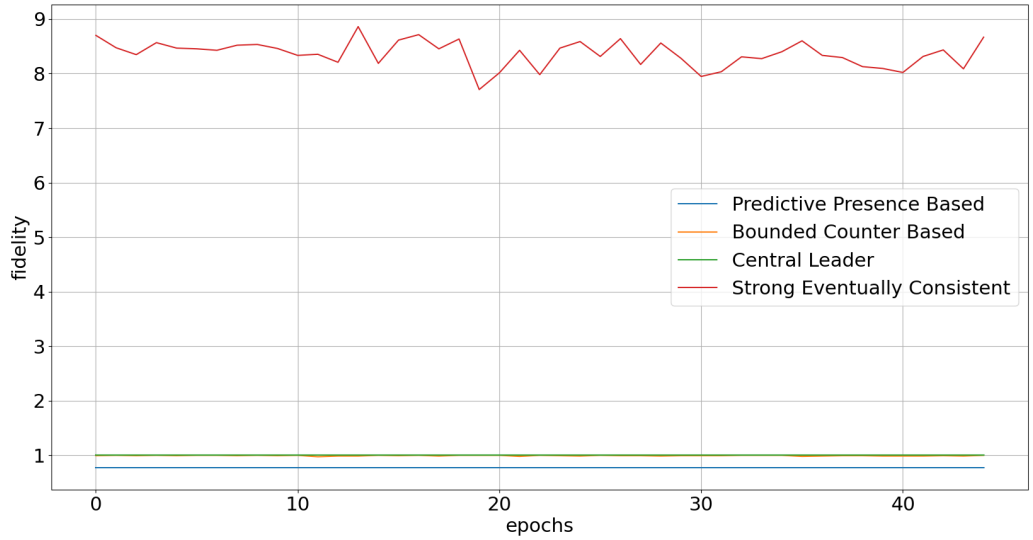

Fig. 7.3.: Fidelity of all limiters

Limiter Algorithm	$Fidelity_{avg}$
Central Leader	1
Strong Eventually Consistent	8.36
Predictive Presence Based	0.77
Bounded Counter Based	0.99

Tab. 7.2.: $Fidelity_{avg}$ for all limiters

By excluding the SEC limiter, Figure 7.4 make the image clearer. The Central Leader is exactly at the prescribed limit. However, the behavior of the Predictive Limiter and the Token Limiter are harder to describe. None of them oversteps the limit, but the BC Limiter has trouble reaching a ratio of 1, while the Predictive Presence Based limiter stay reliably at 77% percent of the limit. We propose some additional experiments to understand their behavior.

7.3.3 Central Leader Limiter

From the analysis of the fidelity and the response time of the Central Leader Limiter, we conclude that it behaves as expected from Section 6.3. The Central Leader Limiter provides strong consistency guarantees with its total order over all operations. As a consequence, our invariant on the spending limit is upheld by the network. However

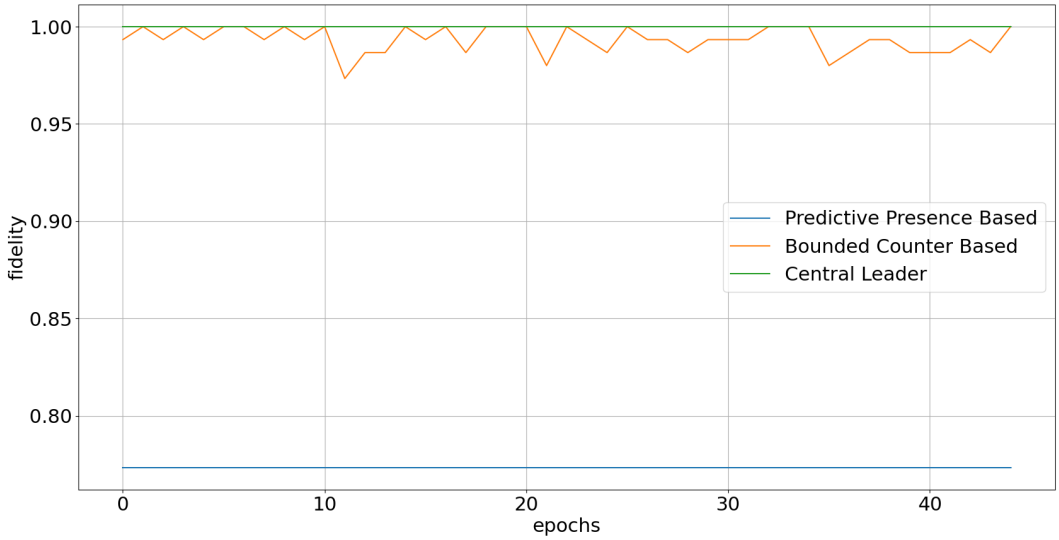

Fig. 7.4.: Fidelity of the limiters, excluding the SEC Limiter

as Figure 7.2 shows, these guarantees come at a high response time cost for most of the requests. There is also a problem of resilience for the network, with the central replica becoming unavailable in the event of a network partition.

These results are also in accordance with the Highly Available and Consistent zones we presented in Section 3.2.2. A strongly consistent limiter can be used in use cases where clients are centralized.

7.3.4 Predictive Limiter

We want to understand why the PPB Limiter is not able to approve 100% of the requests that the global spending limit allows. Which is we present few more runs of the simulation.

Figure 7.5 plots the normalized fidelity of the PPB Limiter in three different simulation setup. For the first two simulations, the parameters are the same as previously, except for the global spending limit that is set at 50 and 200 credit. The last simulation sets the limit at 50 credits and uses a different network topology. In this new topology, we reduce the number of datacenters to one, and we only place 19 antennas on the grid.

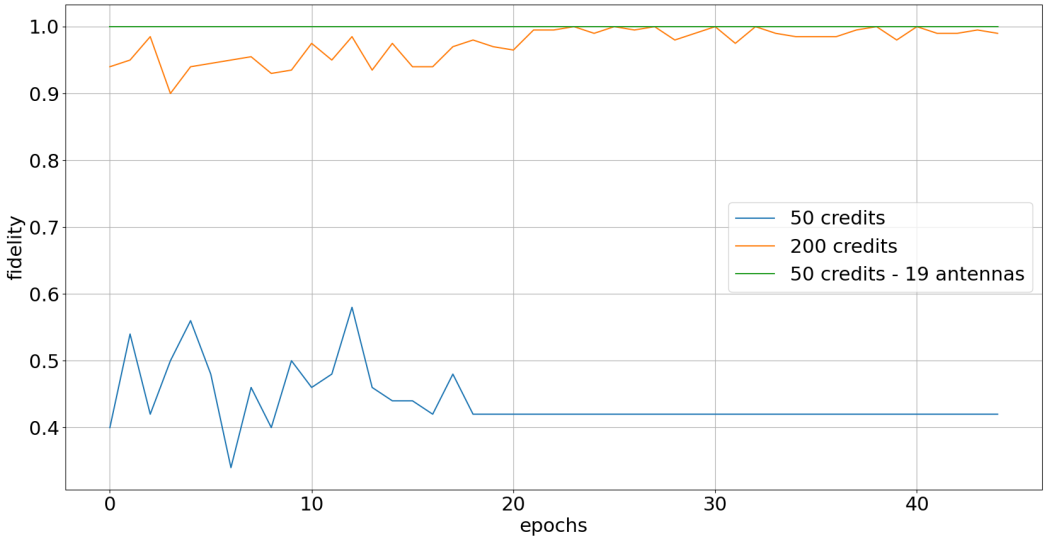

Fig. 7.5.: Fidelity of the Predictive Limiter in different setups

Figure7.3 shows the fidelity ratio for each epoch of the simulation. The $Fidelity_{average}$ is shown in Table 7.3.

Simulation Configuration	$Fidelity_{avg}$
50 credits	0.41
200 credits	0.93
50 credits - 19 antennas	0.95

Tab. 7.3.: $Fidelity_{avg}$ of the Predictive Limiter in different setups

With an increased global limit, the PPB Limiter approves 94% percents of the requests it was supposed to. When the limit is lowered to 50 credits, only 41% of the requests are approved. By lowering the number of antennas, the PPB Limiter reaches 94% of the spending limit.

We conclude from these experiments that the PPB cannot reach a ratio of 1 because the heuristic we use does not allow it to allocate itself the totality of the spending limit. As the number of antenna increases, the proportion each local replica can allocate itself gets smaller. The same is true when the spending limit amount is lowered.

Use cases where limiters of this design would be useful are when mobile devices are restricted to a subset of the antennas in the network. It would have been an

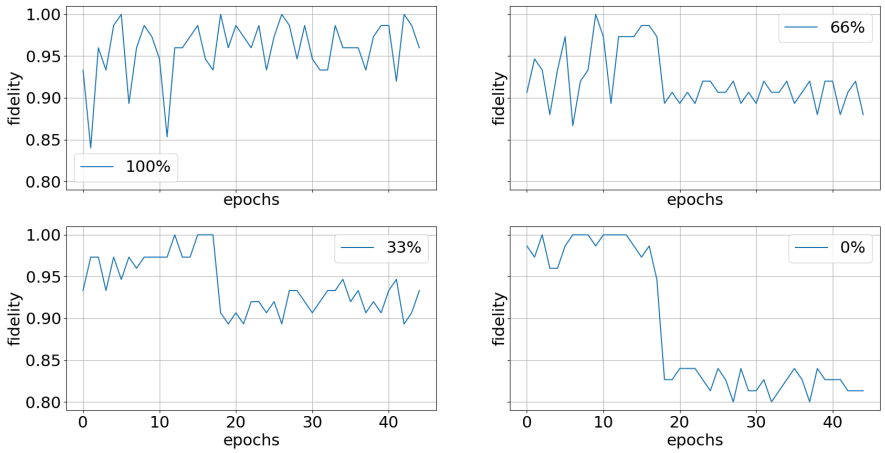

Fig. 7.6.: Bounded Counter based Limiters with a percentage of immobile devices

interesting experiment to define such zones in the simulator. The PPB Limiter could be an effective design with several non-contiguous zone and antennas that are placed far apart from each other. If the latency between all the antennas is short, the Constant Leader would still be effective.

7.3.5 Bounded Counter based Limiter

We want to achieve a $Fidelity_{avg}$ of 1 for the Bounded Counter based Limiter. To understand when this limiter can achieve this level of fidelity, we run several experiments. We lower the global spending limit to 75 to increase contention, and we run it four times, each with a different percentage of mobile devices immobilized at the start of the simulation. These percentages are 0%, 33%, 66% and 100% of devices.

Figure 7.6 shows the results of these four runs, and the ratio they reach at each epoch. Table 7.4 shows the $Fidelity_{avg}$ for the three runs of the simulation and their average response time. Figure 7.7 show how the mobility of devices affects response time.

The percentage of immobile devices affects the $Fidelity_{avg}$. A simulation where all devices are made immobile at the start reaches a greater fidelity than when all devices are mobile. However the fidelity is lower at 33% than it is at 66%. This is also true for the response time.

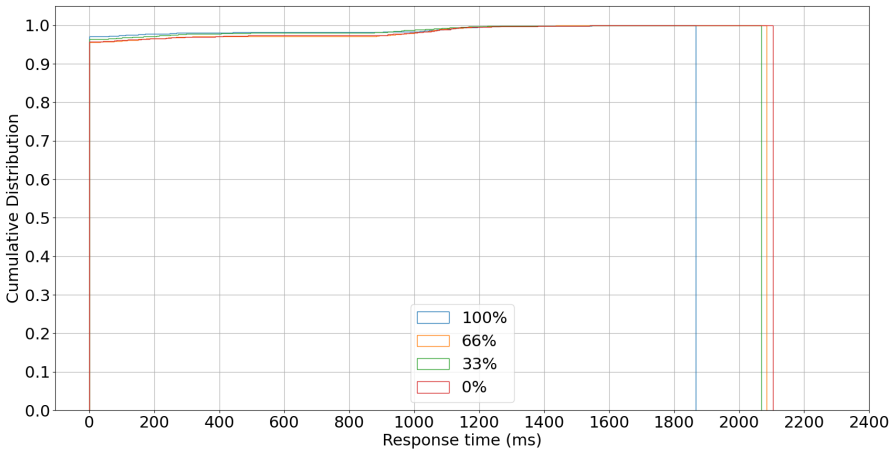

Fig. 7.7.: Bounded Counter based Limiters response time with a percentage of immobile devices

Percentage of immobile devices	$Fidelity_{avg}$	average response time
100%	0.96	23ms
66%	0.92	34ms
33%	0.94	25ms
0%	0.89	33ms

Tab. 7.4.: $Fidelity_{avg}$ of the BC Limiter with different percentages of immobile devices

Reaching this limit should be possible with a greater locality of requests. If all requests take place in a location where a previous request also took place, a token should be available locally. Figure7.6 shows the fidelity of the token limiter at four different levels of token mobility. We can also increase the contention between devices by lowering the global limit to half its previous value.

The outcome of this experiment is that it shows how setting a percentage of immobile devices has its limits here as an approximation. We wanted to ensure that a percentage of requests in the current epoch take place in the same location as during the previous epoch. That way, replicas would be able to repeatedly spend their tokens in the same location and the response time would be shortened.

However, replicas steal from each other as there is no notion of local priority for the use of the tokens. Even if all devices are fixed in a location and it is very likely that a device will make a request locally, when a replica receives a request, it will still send back one token.

Moreover the maximum latency between two servers on the network is longer than the current epoch. This means that any token in transit between two replicas may arrive too late to be used during the current epoch.

Finally, there is a notion of the visibility of operations that must be taken into account. When a local replica needs to token, it consults its local view of the current state of the other replica, and chose a recipient to its request. By the time a message reaches its destination, the token may already have been spent or transferred over to another replica.

The initial distribution of tokens as well as the rules that govern the exchange of tokens between replicas could be adapted to the topology of the network to attempt to reach a $Fidelity_{avg}$ of 1,

7.4 Discussion

During the simulations runs, we showed that two of the limiters behaved as expected. The Central Limiter is highly consistent and enforces the invariant of our global spending limit properly, at the cost of a high response time, and unavailability in the event of a network partition. The SEC Limiter behaves the opposite way. It boasts a very low response time, but the global limit is largely exceeded.

For the two remaining limiters, however, the situation is not as clear. The Bounded Counter based Limiter has a variable response time, but it responds quickly most of the time and guarantees that the limit will not be exceeded. The Predictive Limiter holds the same guarantees.

However, when they are not used in a suitable network these two limiters can enforce a limit on the use of the slice that is lower than what should be allowed. The conditions in which the Token based limiter and the Predictive limiter behave optimally are hard to pin down, and our evaluation has attempted to understand which settings are the most appropriate.

As was already proposed in Section 3.2, there is no single best solution for enforcing a global limit over a geo-distributed system. This section exposed the tradeoffs of different limiters, as we have done for the role of the locality of requests, the topology of the network and the contention over shared resources.

To relate these results to the examples we discussed in Sections 2.3 and 6.7, we can see that only the SEC limiter is suitable for the Critical Slice. For this slice, response

time is mission critical and the fidelity is only a secondary concern. This corresponds to the performance of a limiter whose purpose is the computation of the use of the slice after the fact.

The performance of the CL Limiter verified that it should be privileged for the industrial use case if the latency between the location permits. Otherwise the PPB Limiter had good performances when the distribution of the clients around the antennas is appropriate. The heuristic of the PPB could be adapted to the usage patterns of the different locations.

The fidelity of the BC Limiter during our simulations rarely reached 1. We expected that it could serve the Itinerant Slice especially well because of the migrations of clients and tokens. A lack of time has prevented us from running a simulation where the mobility pattern of the devices adhered strictly to the Itinerant Slice scenario.

8 Conclusion

This thesis studies the consistency requirements of the 5G network. The 5G specification's requirements are too strong, and they would prevent the network from being resilient to network partitions.

Our study of three mobile network procedures from the specification outlined invariants designed to preserve the correctness of the network and consistency semantics that are compatible with high availability.

We went further in our study of the enforcement of a global resource consumption limit to Network Slicing. We assessed the performance of four limiters, at various points of the consistency spectrum, in a simulated 5G deployment.

None of these limiters is an optimal solution to the enforcement of a global limit, but in some specific situations they can prove sufficient. It is in the nature of the tradeoffs we are faced with that the optimal consistency solution depends on the slice's client's requirements. Latency sensitive use cases cannot be served by a limiter with a high response time. If a use case consumes a lot of resources, the operator will filter its use with a limiter that has a good fidelity performance. The same is true if the network has an availability requirement, and the operator is not allowed to refuse client requests when the global limit has not been reached. This imposes further requirements on the fidelity of the limiter.

This highlights the condition necessary to move forward with this work on the consistency requirements of the network. A more precise understanding of the use cases and of the core network is needed. As preparation are made for operators to deploy their core network, they build this precise understanding of its topology. The same is true for the industry as they come forward with new ideas to leverage network slices. This will open the door for some future work, such as

- Adapting the topology of our simulator, or building a large scale core network simulator that is in accordance with the deployment plans of the operators. This will allow for an evaluation of consistency mechanisms that is more realistic and informative than our present attempt.

- Solidifying our hypothesis on how Network Slicing will be used, with a greater variety of use cases brought forward by the industry. From these use cases will flow a new understanding of mobile devices' usage and mobility patterns. From this, consistency mechanisms such as our limiters could be designed around the specificities of their clients.
- Investigating new ideas that have been brought forward as possible innovation for the core network, such as Affirmed Network's proposal of deploying a core network on demand for roaming clients abroad to limit the reliance on the local operator's core network [25]. This concept is easily extensible to 5G and brings new challenges to the management of the user's data, and to the task of keeping it consistent despite the more important latency. Especially if global data is involved such as in this study with Slicing.

The main insight that should be taken away from this thesis is that a consistency model cannot be decreed globally for 5G. Instead a careful analysis of the role of data within the network must guide its designers in outlining the invariants the system relies on to maintain correctness.

Bibliography

[1] 3GPP TS 23.501. “System Architecture for the 5G System, Release 16”. In: (2019) (cit. on pp. 25, 27, 31, 85).

[2] 3GPP TS 23.502. “Procedures for the 5G System, Release 16”. In: (2019) (cit. on pp. 25, 33, 35, 38).

[3] Mustaque Ahamad, Gil Neiger, James E. Burns, Prince Kohli, and Phillip W. Hutto. “Causal memory: definitions, implementation, and programming”. In: 9.1 (Mar. 1995), pp. 37–49 (cit. on p. 17).

[4] Deepthi Devaki Akkoorath, Alejandro Z. Tomsic, Manuel Bravo, et al. “Cure: Strong semantics meets high availability and low latency”. In: Nara, Japan, June 2016, pp. 405–414 (cit. on pp. 20, 21).

[5] NGMN Alliance. “5G Whitepaper, `https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2015/NGMN_5G_White_Paper_V1_0.pdf`”. In: (Oct. 2015) (cit. on pp. 6, 7).

[6] Sérgio Almeida, João Leitão, and Luís Rodrigues. “ChainReaction: A Causal+ Consistent Datastore Based on Chain Replication”. In: *Proceedings of the 8th ACM European Conference on Computer Systems*. EuroSys ’13. Prague, Czech Republic: ACM, 2013, pp. 85–98 (cit. on p. 22).

[7] Nils Aschenbruck, Raphael Ernst, Elmar Gerhards-Padilla, and Matthias Schwamborn. “BonnMotion: A Mobility Scenario Generation and Analysis Tool”. In: *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*. SIMUTools ’10. Torremolinos, Malaga, Spain: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010 (cit. on p. 64).

[8] Peter Bailis, Aaron Davidson, Alan Fekete, et al. “Highly Available Transactions: Virtues and Limitations”. In: 7.3 (Nov. 2013), pp. 181–192 (cit. on p. 42).

[9] Valter Balegas, Diogo Serra, Sergio Duarte, et al. “Extending Eventually Consistent Cloud Databases for Enforcing Numeric Invariants”. In: IEEE, Sept. 2015 (cit. on p. 56).

[10] Hal Berenson, Phil Bernstein, Jim Gray, et al. “A critique of ANSI SQL isolation levels”. In: *ACM SIGMOD Record* 24.2 (1995), pp. 1–10 (cit. on p. 16).

[11] Manuel Bravo, Luís Rodrigues, and Peter Van Roy. “Saturn: a Distributed Metadata Service for Causal Consistency”. en. In: *Proceedings of the Twelfth European Conference on Computer Systems - EuroSys ’17*. Belgrade, Serbia: ACM Press, 2017, pp. 111–126 (cit. on p. 22).

[12] Josh Broch, David A Maltz, David B Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. “A performance comparison of multi-hop wireless ad hoc network routing protocols”. In: *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. 1998, pp. 85–97 (cit. on p. 64).

[13] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N. Calheiros. “Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities”. In: *2009 International Conference on High Performance Computing Simulation*. 2009, pp. 1–11 (cit. on p. 61).

[14] ETSI Mobile Edge Computing, I Initiative, et al. “Mobile-edge computing: introductory technical white paper”. In: *ETSI: Sophia Antipolis, France* (2014), pp. 1–36 (cit. on p. 31).

[15] Jiaqing Du, Calin Iorgulescu, Amitabha Roy, and Willy Zwaenepoel. “GentleRain: Cheap and Scalable Causal Consistency with Physical Clocks”. In: *Proceedings of the ACM Symposium on Cloud Computing* (2014) (cit. on p. 21).

[16] Seth Gilbert and Nancy Lynch. “Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services”. In: 33.2 (2002), pp. 51–59 (cit. on pp. 22, 25).

[17] Alain Girault, Gregor Gössler, Rachid Guerraoui, Jad Hamza, and Dragos-Adrian Seredinschi. “Monotonic Prefix Consistency in Distributed Systems”. In: Springer International Publishing, 2018, pp. 41–57 (cit. on p. 23).

[18] Maurice P Herlihy and Jeannette M Wing. “Linearizability: A correctness condition for concurrent objects”. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 12.3 (1990), pp. 463–492 (cit. on p. 15).

[19] Dongxu Huang, Qi Liu, Qiu Cui, et al. “TiDB: A Raft-Based HTAP Database”. In: *Proc. VLDB Endow.* 13.12 (Aug. 2020), 3072–3084 (cit. on p. 24).

[20] Leslie Lamport. “Time, Clocks, and the Ordering of Events in a Distributed System”. In: *Commun. ACM* 21.7 (July 1978), 558–565 (cit. on p. 17).

[21] Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, and David G. Andersen. “Don’t Settle for Eventual: Scalable Causal Consistency for Wide-Area Storage with COPS”. In: *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. SOSP ’11. Cascais, Portugal: Association for Computing Machinery, 2011, 401–416 (cit. on p. 20).

[22] Frans Ojala, Ashwin Rao, Hannu Flinck, and Sasu Tarkoma. “NoSQL stores for coreless mobile networks”. In: *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*. IEEE, 2017 (cit. on pp. 24, 25).

[23] Matthieu Perrin, Matoula Petrolia, Achour Mostéfaoui, and Claude Jard. “Consistent shared data types: Beyond memory”. PhD thesis. Université de Nantes, 2014 (cit. on p. 17).

[24] Yasushi Saito and Marc Shapiro. “Optimistic replication”. In: 37.1 (Mar. 2005), pp. 42–81 (cit. on p. 19).

[25] Amazon Web Services. “Whitepaper: Carrier-Grade Mobile Packet Core Network on AWS”. In: (2019) (cit. on p. 76).

[26] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. “Conflict-Free Replicated Data Types”. In: Springer Berlin Heidelberg, 2011, pp. 386–400 (cit. on pp. 19, 20).

[27] Jonathan Sid-Otmane, Sofiane Imadali, Frédéric Martelli, and Marc Shapiro. “Data Consistency in the 5G Specification”. In: *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. 2020, pp. 110–117 (cit. on p. vii).

[28] *SliceCap sources*. URL: `github.com/SmallEndian/SliceCap` (cit. on p. 61).

[29] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. “EdgeCloudSim: An environment for performance evaluation of Edge Computing systems”. In: *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. 2017, pp. 39–44 (cit. on pp. 4, 61, 64).

[30] Andrew S Tanenbaum and Martin Van Steen. “Distributed systems principles and paradigms. 2002”. In: *Cited in* (2016), p. 326 (cit. on p. 18).

[31] Ilyas Toumlilt, Pierre Sutra, and Marc Shapiro. “Highly-Available and Consistent Group Collaboration at the Edge with Colony”. In: *Middleware 2021: 22nd International Middleware Conference*. Québec (online), Canada: ACM, Dec. 2021 (cit. on p. 23).

[32] J. Turner. “New directions in communications (or which way to the information age?)” In: *IEEE Communications Magazine* 24.10 (1986), pp. 8–15 (cit. on p. 49).

[33] International Telecommunication Union. “ITU-T Rec. I.371 - Traffic control and congestion control in B-ISDN”. In: (2004) (cit. on p. 49).

[34] ETSI GS NFV 003 V1.4.1. “Terminology for Main Concepts in NFV”. In: (2018) (cit. on p. 5).

[35] Paolo Viotti and Marko Vukolić. “Consistency in non-transactional distributed storage systems”. In: *ACM Computing Surveys (CSUR)* 49.1 (2016), pp. 1–34 (cit. on p. 18).

[36] Mathias Weber, Annette Bieniusa, and Arnd Poetzsch-Heffter. “Access control for weakly consistent replicated information systems”. In: *International Workshop on Security and Trust Management*. Springer. 2016, pp. 82–97 (cit. on p. 24).

Acronyms

A

3GPP 3rd Generation Partnership Project 1, 25

ACID Atomicity, Consistency, Isolation, and Durability 23–25

AF These are application specific functions running within the 5G infrstructure. This allows the to interact with the Policy Control Function (PCF) or even to influence routing decisions. 28

AMF The is the central function that manages the User Equipment (UE). It receives requests directly from the UE and makes decisions accordingly or forwards them to different Virtual Network Functions (VNFs) 27, 29–34, 36, 37, 39, 43, 44, 83, 84, 86

AUSF This is a server that authenticates the UE when it first connects to the network. It maitains security information in its own section of the Unified Data Management (UDM) and installs a security context on the Access and Mobility Function (AMF)s and the UE so that all future communications with the network are encrypted. 25, 27, 33, 39

COTS Commercial Off The Shelf 2, 5, 27

CRDT Conflict-free Replicated Data Type 17, 20, 45, 56

DN The Data Network (DN) or Packet Data Network (PDN) is the network targeted by the UE, for data or voice services, the Internet for instance. 6, 26, 28, 34, 36, 39, 44, 83

DNN This is an identifier for the Data Network the subscriber is trying to reach. 31, 84, 85

eMBB enhanced Mobile BroadBand 7–9

GPRS General Packet Radio Service 1

GSM Global System for Mobile Communications 1

HAC Highly Available and Consistent 24, 45, 46

IoT Internet of Things 2, 8

IP Internet Protocol 1

LADN A Data Network Name (DNN) that may only be accessed while in physical proximity. Every AMF knows all the nearby Local Area Data Network (LADN)s. 31, 34, 39, 84

LTE Long-Term Evolution 1, 7, 48

M2M Machine to Machine 1, 10

MCX These are information concerning the degree of priority that this UE should be afforded. 31

MEC Mobile Edge Computing 31

mMTC massive Machine-Type Communications 7, 8

MNO Mobile Network Operator 1, 2, 10

Mobility Restrictions A set of restrictions applied to the UE. They restrict mobility handling procedues and service access (e.g. access closed to groups, forbidden areas). 31

NF Network Function 5, 13, 27

NFV Network Function Virtualization 2, 5

NSSAI Network Slice Selection Assistance Information 32, 85

NSSF The Network Slice Selection Function (NSSF) is a repository of information on slices. It is intended to advertise their features when several are availble for the user. 28, 84

PCF This VNF ensures that all policies (Quality of Service (QoS), Spending Limits, allowed types of connection) regarding a subscriber, an UE or a PDU Session are known and kept up to date. 28, 30, 34, 36, 37, 46, 48, 49, 83

PDN Describes a Data Network where pakets are used instead of circuit switching. Most common examples are IP networks. 26, 28, 31, 32, 85, 86

PDU Packet Data Unit (PDU) a unit of information that is exchanged between the UE and any Packet Data Network (PDN) that the user is subscribed to (i.e. IPv4, IPv6, Ethernet). Packet Data Unit (PDU) connectivity is the service that provides the exchange of PDUs between the UE and a PDN identified by its DNN by establishing a *PDU Session*.[1] 30–32, 35, 43, 46, 85, 86, 93

PDU-Session A PDU is a logical connection set up between the UE and a target PDN. 27, 30–32, 86

PEI A unique identifier for the device through which the UE is reaching the network *e.g.* the IMEI 31

PLMN Public Land Mobile Network 9, 10

Priority Flag This is a flag set in the header of the session establishment messages when appropriate according to the MCX priorities. 31

PSA PDU Session Anchor 28, 36, 86

QoE Quality of Experience 61

QoS This contains all the information on the performance of the connection. 6, 7, 9, 28, 32, 36, 47, 84

RAN This function handles the radio access technology and the communication between the UE and the network. 5, 26, 28, 37

RTT Round Trip Time 55, 65

S-NSSAI This is the identifier of a slice selected through the Network Slice Selection Assistance Information (NSSAI) 32, 44

SDM This is all the information pertaining to a Subscriber. 32, 36

SDN Software Defined Networking 1, 2, 5

SEC Strong Eventual Consistency 67–69, 73, 87, 93

SLA Service Level Agreement 6, 8, 48

Slicing Change Indicator When this value changes it signals to the VNF that the subscription data for network slicing changed and the UE configuration must be updated. 32

SLO Service Level Objective 2, 6

SM Context This contains all the information pertaining to current PDU sessions and to the set up of future sessions. 32

SMF This is the VNF in charge of setting up and managing one or several Packet Data Unit Session (PDU-Session). It will not change during the lifetime of the session. 27, 30, 34, 36, 37, 39, 40, 44, 45

Steering of Roaming The information used by the UE to know which foreign networks it should connect to while abroad. 32

SUCI This is a single use identifier for the client that can be broadcast in clear to the AMF while still preserving the anonimity of the subscriber. 32, 39

SUPI Subscription Permanent Identifier 33, 34

Tunnel Info Contains the information necessary to route packets to the correct User Plane Function (UPF), PDN and UE. 32

UDM This is the unified interface to 5G's data store. It allows 5G core network components to query, subscribe to, and modify values. 13, 14, 25, 27, 30, 33, 34, 36, 37, 39–41, 44, 83

UE This is the function embedded in the device that allows the subscriber to access the network. x, 6, 24–34, 36, 37, 39, 40, 42–46, 48, 83–86, 93

UE-context User Equipment Context 24, 25, 30, 33, 37, 39, 43, 44

UMTS Universal Mobile Telecommunications System 1

UPF This is the component that carries data packets between the UE and the PDN, either directly or through Intermediate UPF (I-UPF). The PDU Session Anchor (PSA) is the closest to the PDN and is chosen for the lifetime of the session. 28, 30, 32, 34, 36, 37, 39, 45, 86

URLLC Ultra-Reliable and Low-Latency Communications 7

VM Virtual Machine 2

VNF Virtual Network Function 1, 5, 6, 25–28, 32, 40–42, 83–86

VPN Virtual Private Network 10

Limiters Algorithms

B

In this section we present teh limiter designs of Section 6 in the form of pseudo-code. As in the simulator shown in Section 7, limiters send and receive messages from themselves and from the clients. This is why we present the pseudocode in the form of the contents of a message loop, where the message that has just been received is treated with a switch statement.

The types of messages limiters exchange and receive are as follows.

- Init: is the message sent during the initialization of the limiters. This is when limiters first set up their local state.
- Request: is the message sent by a client to request the use of the slice from the limiter.
- Epoch: is a message recieved periodically at the end of an epoch. After reception of this message, the spending limit is reset.
- Update: is a message sent between limiters to inform them when an update has taken place.
- Solicit: for the Token based limiter, a replica needs to be able to request a token from another replica. This is the purpose of this message.

The Strong Eventual Consistency (SEC) limiter from Section 6.4 is shown in Algorithm 2.

The Central Leader limiter from Section 6.3 is shown in Algorithm 1.

The Token based limiter from Section 6.5 is shown in Algorithm 3. This limiter relies on tokens to be able to function, some of these function are a bit to hard to be easily readable. This is why we extracted them in their own listings.

For the Predictive limiter from Section 6.6 in Algorithm 4 we assume that the antenna and the slice offer two key informations to the limiter: the number of devices in range who are subscribed to the slice, and the total number of devices that are subscribed to the slice.

```
Input: message
switch message.type do
    case Init do
        leader = message.leader_id;
        if leader == local_id then
            credit = cap;
        end
        break;
    end
    case Request do
        if leader != local_id then
            /* Requests are simply passed along to the leader   */
            send(leader, message);
        else
            if credit > 0 then
                credit -= 1;
                approve(message.task);
            else
                deny(message.task);
            end
        end
        break;
    end
    case Epoch do
        credit = cap;
        break;
    end
end
```

Algorithm 1: Central Leader

```
Input: message
switch message.type do
    case Init do
        epoch = 0;
        spent[] ← ∅;
        break;
    end
    case Request do
        if sum(spent) < cap then
            spent[local_id] += 1;
            approve(mesage.task);
            foreach Limiter other do
                /* Inform all other replicas that some of the
                   limit has been spent                          */
                send(other, spent.copy);
            end
        else
            deny(message.task);
        end
        break;
    end
    case Update do
        /* Another limiter has approved a task                   */
        if epoch == epoch.message then
            for i in spent.length do
                spent[i] = max(spent[i], message.spent[i]);
            end
        end
        break;
    end
    case Epoch do
        epoch++;
        spent ← ∅;
        break;
    end
end
```

Algorithm 2: Strong Eventually Consistent Limiter

```
Input: message
switch message.type do
    case Init do
        tokens[][] ← ∅;
        spent[] ← ∅;
        epoch = 0;
        /* At initialization time, a predetermines replica recieves
           all the tokens.                                          */
        tokens[message.leader][message.leader] = cap;
    end
    case Request do
        if compute_local_credit(tokens, spent, local_id) > 0 then
            spent[local_id] += 1;
            approve(task);
        else
            if global_credit(tokens, spent) > 0 then
                send(select_replica(tokens, spent), Solicit);
            end
            deny(message.task)
        end
        break;
    end
    case Solicit do
        /* Another replica requires a token                         */
        if compute_local_credit(tokens, spent, local_id) > 0 then
            tokens[message.sender][local_id] += 1;
            send(message.sender, Update, tokens.copy(), spent.copy(), true);
        else
            deny(message.task);
            send(message.sender, Update, tokens.copy(), spent.copy(), false);
        end
        break;
    end
    case Update do
        /* Update our local state to reflect changes from other
           replicas                                                 */
        for i in tokens.length do
            for j in tokens[j].length do
                tokens[i][j] = max(tokens[i][j], message.tokens[i][j]);
            end
            if message.epoch == epoch then
                spent[i] = max(spent[i], update.tokens[i]);
            end
            if message.boolean  compute_local_credit(tokens, spent, local_id) > 0
             then
                spent[local_id] += 1;
                approve(message.task);
            end
        end
        break;
    end
    case Epoch do
        epoch += 1;
        spent ← ∅; end
    end
```

Algorithm 3: Tokens based limiter

```
Input: message
switch message.type do
    case Init do
        subsrcribed_devices =
         message.slice.get_subscribed_devices_count();
    end
    case Epoch do
        local_cap = (antenna.get_local_devices_count() /
         subscribed_devices) * cap;
        break;
    end
    case Request do
        if local_cap > 0 then
            approve(message.task);
            local_cap -= 1;
        else
            deny(message.task);
        end
        break;
    end
end
```

Algorithm 4: Predictive Limiter

```
Input: tokens, spent, id
Output: credits
/* How many unspent tokens does limiter id possess          */
1  credit = 0;
/* Calculate the sum of all tokens that have been sent to us
   */
2  for i in tokens[id].length do
3      credit += tokens[id][i];
4  end
/* Remove the credits that we have sent to another limiter  */
5  for i in tokens.length do
6      if id != i then
7          credit -= tokens[i][id];
8      end
9  end
/* Remove the credits that have been spent                  */
10 credit -= spent[id]
11 return credit;
```

Function compute_local_credit(tokens, spent, id)

```
Input: tokens, spent
Output: global_credits
/* How many unspent tokens remain globally                    */
credit = 0;
/* How many unspent tokens are there in every other limiter   */
for i in tokens.length do
    credit += compute_local_credit(tokens, spent, i);
end
return credit;
```

Function compute_global_credit(tokens, spent)

```
Input: tokens, spent
Output: replica_id
/* Find a limiter that still owns some credits                */
replica_id = 0;
current_max=0;
for i in tokens.length do
    current = compute_local_credit(tokens, spent, i);
    if current > current_max then
        current_max = current;
        replica_id = i;
    end
    return replica_id;
end
return id;
```

Function select_replica(tokens, spent)

List of Figures

List of Tables