# Reducing latency and jitter in 5G radio access networks

Flavien Ronteix–Jacquet

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS-DE-LA-LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 601
*Mathématiques et Sciences et Technologies de l'Information et de la Communication*
Spécialité : *Informatique*

Par
**Flavien RONTEIX- -JACQUET**

## Réduction de la Latence et de la Gigue dans le Réseau d'Accès Radio 5G

Reducing Latency and Jitter in the 5G Radio Access Network

**Thèse présentée et soutenue à l'IMT Atlantique à Rennes, le 13 Décembre 2022**
**Unité de recherche : Adopnet / IRISA**
**Thèse N° : 2022IMTA0324**

### Rapporteurs avant soutenance :

| | |
|---|---|
| Philippe Martins | Professeur, Télécom ParisTech |
| Thierry Turletti | Directeur de Recherche, INRIA |

### Composition du Jury :

| | | |
|---|---|---|
| Président : | André-Luc Beylot | Professeur des Universités, ENSEEIHT |
| Examinateurs : | Cédric Adjih | Chargé de Recherche, INRIA |
| | Philippe Martins | Professeur, Télécom ParisTech |
| | Thierry Turletti | Directeur de Recherche, INRIA |
| Directeur de thèse : | Xavier Lagrange | Professeur, IMT Atlantique |
| Co-Dir. de thèse : | Isabelle Hamchaoui | Ingénieur de recherche, Orange Innovation |
| Encadrant de thèse : | Alexandre Ferrieux | Ingénieur de recherche, Orange Innovation |

### Invité(s) :

| | |
|---|---|
| Stéphane Tuffin | Ingénieur de recherche, Orange Innovation |

# Remerciements

*"Peut-on écrire une thèse sans mourir d'ennui ou devenir à moitié fou ? oui, et mieux encore, il faut vivre la thèse comme un défi"*[1], et cette thèse a été un véritable défi de trois ans mais surtout, le travail intellectuel le plus stimulant de ma vie.

Ce travail n'aurait jamais pu exister sans mes encadrants, Alexandre Ferrieux et Isabelle Hamchaoui auprès de qui j'ai énormément appris depuis trois ans et demi que nous travaillons ensemble. Aussi, je ne remercierai jamais assez mon directeur de thèse, le Pr. Xavier Lagrange pour sa confiance, son suivi et ses excellents conseils tout au long de cette thèse. Je souhaiterais également remercier mes rapporteurs de thèse pour leur temps et leurs commentaires pertinents et mon jury de thèse d'avoir accepté de juger mon travail.

Je tiens à remercier tous mes collègues d'Orange (ex-CNET): les équipes ITEQ et AMOX, la bibliothèque technique, la section syndicale, Ziad, Benoît, Olivier pour ces discussions passionnantes que nous avons eues ensemble à l'occasion d'un café, d'un repas.

Également merci à Laurent Thomas et à toute la communauté OAI de faire vivre ce superbe projet qu'est OpenAirInterface.

De manière plus personnelle, merci à l'émission "C'est pas sorcier" pour mon goût pour les Sciences depuis 20 ans. Dany et Raz pour votre contenu qui m'a déterminé à avancer. A tous mes camarades, Jean, Dani, la section du Trégor, Camille M., les fédés des jeunes bretons... la lutte continue. Merci à mes amis qui m'ont soutenus (et supportés) ces dernières années: Lorin, Anastasia, Maxime, Mélanie, Stéphane, Rémi, Kimi, Camille B., Aurélien, Anne-Marie, Clément et Kat. On se revoit bientôt.

Merci à ma famille bien évidemment, qui ne comprennent plus tout à fait ce que je fais depuis 10 ans mais sans qui je ne serais pas la personne qui présente cette thèse.

Et à Danielle. Toute ta vie, tu m'as tellement souhaité d'aller le plus loin possible dans mes études et mes recherches. Tu n'es malheureusement plus là pour voir ce souhait réalisé. Cette thèse t'est dédiée...

1. "Comment écrire sa thèse", Umberto Eco (2016).

# Résumé en Français

Le développement des réseaux cellulaires LTE (4G) a permis au cours de la précédente décennie l'émergence de services demandeurs en débit (ex. la vidéo) pour les utilisateurs mobiles. Cependant, les réseaux LTE présentent des caractéristiques de latence qui ne sont pas compatibles avec un certain nombre de services, comme le jeu vidéo dans le cloud. La latence est le temps nécessaire à un paquet de données pour traverser un segment du réseau. La latence est particulièrement importante dans le réseau d'accès cellulaire (Radio Access Network, RAN) comme le relève la littérature et l'expérience terrain des opérateurs. La latence minimale d'aller-retour (AR) avec la technologie LTE est de l'ordre de 20 ms, beaucoup plus que dans l'accès fixe où il est courant d'avoir des latences inférieures à 10 ms. Surtout, la latence du réseau d'accès en réalité dépasse régulièrement les 50 ms jusqu'à plusieurs centaines lorsque le point d'accès est congestionné. Les services nécessitant de la faible latence sont aussi souvent sensibles à la variation de cette latence dans le temps, ce qui est appelée la gigue. La gigue dans le RAN LTE se révèle aussi très importante, principalement dû aux changements dans le temps des caractéristiques du lien radio, les retransmissions pour pallier les pertes, et à la concurrence d'accès aux ressources radio des différents utilisateurs.

La dernière génération de technologie cellulaire 5G [1] a parmi ses objectifs la réduction de la latence pour tendre vers la milliseconde dans le RAN. Pour cela, certains mécanismes d'adaptation aux conditions radio et de récupération des pertes ont été améliorés. Surtout, il a été introduit une nouvelle méthode de gestion des flux de bout en bout pour contrôler et garantir des latences faibles, basée sur du découpage du réseau (slicing), de la différentiation des flux et la gestion active de qualité de service (QoS). Ces mécanismes de gestion active de la latence ne sont actuellement pas utilisés pour les flux Internet qui sont généralement transmis dans un mode "Best-Effort" (au mieux). Or, il s'agit du mode de transmission par défaut des flux utilisateur dans le réseau. De ce fait, la latence minimale dans le RAN pour les utilisateurs une fois que la 5G Standalone sera déployé ne devrait être réduite que de l'ordre de 10 ms, tout en ne garantissant pas plus qu'en LTE cette latence minimale au cours du temps.

L'objectif de ce travail de thèse est de réduire la latence et la gigue introduite par le

réseau d'accès cellulaire, LTE et 5G pour les transmissions de flux Internet.

La méthode utilisée repose sur l'utilisation d'une plateforme d'expérimentation en laboratoire (voir Chapitre 2) composée de terminaux du commerce et d'une station de base (BS) LTE/5G issue du projet open-source OpenAirInterface[2] [2]. Cette méthode permet de contrôler l'environnement radio (avec un canal radio optimal, sans variations, sans perturbations externes) et les latences du réseau tout en étant représentatif des latences utilisateur avec des terminaux du commerce et une station de base de recherche. L'étude des latences se concentre sur le RAN du côté de l'opérateur du réseau ; le cœur réseau et les terminaux sont considérés comme des boîtes noires. À l'inverse, le code source de la station de base utilisée est ouvert, ce qui permet une analyse poussée des mécanismes des protocoles radio. Les latences sont mesurées par paquet à l'aide de 2 captures réseaux simultanées avec une synchronisation des horloges à l'entrée et à la sortie du RAN (méthode passive de mesure contrairement à une méthode active qui utilise un flux pour mesurer la latence). Cette méthode permet d'obtenir un suivi fin de l'évolution de la latence au cours du temps avec une précision sous la milliseconde, mais cette méthode ne permet pas d'analyser la composition interne de la latence.

La première contribution de cette thèse est la proposition d'une nouvelle méthode de mesure des latences internes au RAN, LatSeq (voir Chapitre 3). Nous définissons la latence interne comme la succession de délais subit par un paquet de données dans le système de la station de base. Nous modélisons la succession des délais comme un graphe dont les arêtes correspondent à des segments du trajet du paquet, les sommets comme un lien entre 2 segments (par exemple entre 2 couches protocolaires). LatSeq est implémenté dans la station de base (BS) sous la forme de sondes dans le code qui observent le passage des paquets entre les différents segments. Ces sondes sont conçues pour avoir le minimum d'impact possible en termes de délais et de coût de calcul sur le système observé. L'ensemble des traces produites par les sondes sont écrites au fur et à mesure du fonctionnement du RAN dans un fichier de traces. Le trajet des paquets au sein de la BS sont alors générés par un algorithme de reconstruction de graphe. Les sondes fonctionnent en temps-réel, mais la reconstruction est un processus plus lourd en calcul qui est fait a posteriori. En sortie, l'outil fournit une représentation graphique des latences (voir Fig. 3.10a) par paquet à un niveau de grain qui peut aller jusqu'à la fonction logicielle avec une précision de l'ordre de la microseconde. Une évaluation a été réalisée pour montrer le faible impact de la méthode sur le fonctionnement de la station de base

2. `https://openairinterface.org/`

(16 ns par mesure) et pour montrer la précision de la mesure de la latence. LatSeq (pour Latency Sequence analysis) a fait l'objet d'une publication en conférence internationale [3], d'une publication du code de l'outil en open-source [3] et d'une contribution au projet OpenAirInterface [4].

Les principaux résultats d'expérimentations sur la plateforme expérimentale équipée de Latseq montrent que : 1) la voie descendante (du réseau vers les terminaux) est la source la plus importante de délais lorsque la cellule est chargée par plusieurs flux concurrents (phénomène du "bufferbloat") ; 2) le délai subit par un paquet sur la voie remontante (des terminaux vers le réseau) dans la BS est toujours faible, même si les retransmissions, la segmentation et le "in-order delivery" RLC peuvent en générer ; 3) la voie remontante représente la principale source de gigue du RAN et une latence plus importante que celle de la voie descendante quand la cellule est vide.

L'analyse des latences internes montre que le mécanisme d'accès au canal radio de la voie remontante est responsable de la variation importante de la latence d'un paquet à un autre et de l'apparition de rafales de transmission dans le profile de transmission du trafic [4] (voir Fig. 4.10a). En effet, comme pour la voie descendante, l'allocation des ressources radio est réalisée par l'ordonnanceur MAC de la station de base, par contre à la différence de la voie descendante, la connaissance de l'état des buffers de transmission et des besoins en transmission sont répartis entre les terminaux. Pour résoudre ce problème dû à l'architecture centralisée des cellules radio, les normes LTE et 5G définissent : Le "Scheduling Request" (SR) qui permet à un terminal de demande un accès au canal de la voie remontante ; Le "Buffer Status Report" (BSR) qui permet à un terminal de transmettre à la station de base l'état d'occupation des buffers de transmission ; Le "Uplink Grant" (UG) qui est transmis par la station de base aux terminaux via le canal de contrôle (DCI) et qui indique les ressources radio allouées à un terminal ainsi que le codage et la modulation à utiliser. Le processus est le suivant (voir Fig. 4.15): Le terminal a des données à transmettre et demande donc à la BS des ressources radio avec un SR. La station de base prend en compte cette demande en envoyant un UG au terminal lui allouant des ressources radio sur la voie remontante. Puis le terminal transmet des données sur les ressources radio qui lui ont été réservées et y ajoute un BSR pour indiquer la quantité de données encore à transmettre. L'algorithme d'ordonnancement va prendre en compte cet état du buffer pour allouer la quantité de ressources radio nécessaire pour vider le buffer

3. `https://github.com/Orange-OpenSource/LatSeq`
4. `https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/latseq`

de transmission. C'est le mécanisme de demande explicite d'accès et la connaissance partielle par la station de base de l'état des buffers des terminaux qui génèrent des délais d'accès et des variations dans la latence d'un paquet à l'autre. Il s'agit de cette dernière observation, corrélée avec des observations dans un réseau cellulaire commercial d'Orange (voir paragraphe 4.2.2) et des observations issues de la littérature (voir paragraphe 4.2.6) qui justifient le reste de ce travail de thèse sur le besoin de réduire la gigue et la latence de la voie remontante dans le RAN en travaillant sur le mécanisme d'accès et d'allocation des ressources radio.

La principale contribution de cette thèse [5] est une solution au problème de l'estimation par la station de base (BS) des besoins en transmission des terminaux (voir Chapitre 6) dénommé "Enhanced-BSR". Pour cela, il est proposé d'implémenter un modèle d'estimation au niveau de la BS qui utilise les rapports BSR, l'estimation du débit de transmission du terminal et le taux d'utilisation des blocs de transport (TB) pour fournir une prédiction en continu de l'occupation du buffer de transmission du terminal. Cette information est ensuite utilisée par l'algorithme d'ordonnancement pour répartir les ressources entre les terminaux. Le modèle d'estimation linéaire utilisé s'adapte en continu pour garantir une utilisation des ressources radio allouée à plus de 80%. Le modèle assez simple qui est utilisé permet d'obtenir une bonne estimation des besoins en transmission d'un terminal dont le trafic est régulier avec un débit source qui varie peu à des courtes échelles de temps. Cela correspond typiquement à un transfert TCP, du cloud gaming ou de la vidéo. Les avantages majeurs de cette solution, c'est qu'elle est compatible avec les normes LTE et 5G actuelles, qu'elle ne nécessite pas de modifications de l'algorithme d'ordonnancement ou des terminaux. Une limitation importante est le gaspillage de ressources radio lorsque le modèle surestime les besoins en transmission du terminal.

Une évaluation de la solution montre une nette diminution de la gigue ainsi que de la latence remontante en particulier pour les transferts longs et à débit constant passant d'un délai AR de 52 à 19 ms et d'une gigue de 10 ms au lieu de 47 ms (voir Fig. 6.4). La latence et la gigue sont aussi diminuées pour un trafic non déterministe en rafale, mais occasionne une sous-utilisation des ressources radio allouées. Une conséquence intéressante de la réduction de la gigue et la latence de la voie remontante est à observer sur les trafics TCP descendant (par exemple pour un téléchargement ou le chargement d'une page Web). En effet, TCP est un protocole connecté, par lequel le transfert des données se fait par une voie (par exemple la voie descendante pour un téléchargement), et les données sont acquittées en utilisant la voie inverse (la voie remontante pour un

téléchargement). L'acquittement est utilisé pour garantir la transmission des données, mais aussi pour contrôler la congestion en adaptant le débit de transmission aux capacités du réseau. Les algorithmes de contrôle de congestion TCP les plus utilisés sur internet estiment la capacité du réseau à partir des pertes et de la latence. Nous avons observé une amélioration du débit réalisé et une meilleure stabilité du débit de transfert lorsque notre solution est implémentée (voir le débit en Fig. 6.7c comparé au débit avant avec la Fig. 6.7a). Cette expérimentation montre que la réduction de la gigue et de la latence remontante n'est pas seulement bénéfique pour les services sensibles à la latence, mais aussi pour l'augmentation des débits et pour une meilleure utilisation des ressources [4] radio dans le RAN par les trafics TCP. Or, la sous-utilisation des ressources radio par TCP dans les réseaux cellulaires est un problème souvent évoqué dans la littérature ( 70% en LTE, 40% en 5G [6]) auquel de nombreuses solutions ont été proposées. Enhanced-BSR est une nouvelle approche au problème qui n'a pas recours à une modification de TCP, des applications ou de l'architecture du réseau.

Les perspectives ouvertes par cette thèse sont nombreuses. L'outil LatSeq peut être développé pour répondre à des tâches de supervision et de débogage de la station de base dans un contexte de réseau radio logiciel (SD-RAN). Enhanced-BSR a déjà démontré sur un ensemble d'expérimentations sa capacité à réduire la latence et la gigue sur la voie remontante dans le réseau d'accès cellulaire LTE. Le modèle d'estimation utilisé pourrait être amélioré en utilisant d'autres algorithmes d'estimation basés sur de l'apprentissage automatique des profils de trafic. L'adaptation de l'algorithme d'ordonnancement dynamique de la voie remontante aux prédictions du modèle Enhanced-BSR est aussi une piste intéressante pour réduire la latence et obtenir des meilleures performances de débit dans le réseau 5G tout en garantissant une équité d'accès aux terminaux et une diminution de la consommation énergétique.

# Table of Contents

**Introduction** **1**

Motivation . . . 1

Thesis Contributions . . . 4

Dissertation outline and organization . . . 6

**1 Background** **7**

1.1 5G and Cellular Networks . . . 8

1.1.1 Cellular network architecture . . . 9

1.1.2 Radio Access Network . . . 12

1.1.3 Radio resource allocation . . . 17

1.2 Internet Protocol Suite . . . 26

1.2.1 Transmission Control Protocol (TCP) . . . 26

1.2.2 Congestion control . . . 26

1.2.3 Quic . . . 27

1.3 Latency in Networks and Mitigation . . . 29

1.3.1 Definition of latency . . . 29

1.3.2 Bufferbloat . . . 32

1.3.3 Latency Mitigation in 5G . . . 36

**2 Research Method** **43**

2.1 Objectives . . . 44

2.2 Lab testbed with OpenAirInterface . . . 46

2.2.1 Comparison of RAN solutions . . . 46

2.2.2 Mobile network core . . . 48

2.2.3 Radio Access Network testbed . . . 49

2.3 Traffic generation . . . 52

2.4 Analysis tools and methodologies . . . 55

2.4.1 Active RTT measurement . . . 55

2.4.2 Passive measurements . . . 56

2.4.3 Network captures synchronization . . . 58
2.5 Conclusion . . . 60

**3 LatSeq** **61**
3.1 Motivation . . . 63
3.2 Definitions . . . 64
3.2.1 Internal latency . . . 64
3.2.2 Packet fingerprint . . . 66
3.2.3 Packet journey . . . 68
3.3 Implementation . . . 70
3.3.1 Real-time measurement of packet fingerprint . . . 71
3.3.2 Measurement points in the LTE stack . . . 76
3.3.3 Internal latency analysis . . . 79
3.4 Evaluation . . . 84
3.4.1 Unitary measurement point generation time . . . 85
3.4.2 Memory usage . . . 86
3.4.3 In-running global performance . . . 86
3.5 Related work . . . 88
3.6 Showcasing LatSeq: TCP ACK packet bursting in uplink . . . 89
3.7 Contribution to open source community . . . 91
3.8 Conclusion . . . 92

**4 The Causes of Latency in the Radio Access Networks** **93**
4.1 RAN Latency Characterization in the Lab Testbed . . . 95
4.1.1 Baseline RAN RTT with Ping . . . 96
4.1.2 Connection establishment, RTT and HTTP/S transfer . . . 101
4.1.3 Exhibiting bufferbloat . . . 104
4.2 Uplink Segment as a Source of Latency and Jitter . . . 108
4.2.1 Uplink transfer experiment . . . 108
4.2.2 Uplink latency and jitter in a commercial RAN . . . 110
4.2.3 Analysis of the uplink radio resource allocation scheme . . . 114
4.2.4 Influence of RAN configurations . . . 118
4.2.5 On the buffer knowledge . . . 123
4.2.6 Related observations . . . 124
4.3 Conclusion . . . 128

**5 Why Tackle the Uplink-Channel Latency and Jitter ? 129**

**6 Enhanced-BSR 137**

6.1 Needs of an Improved Buffer Estimation Method . . . 139
6.1.1 Context . . . 139
6.1.2 Our approach . . . 140
6.1.3 Related work . . . 140
6.2 Design and Implementation . . . 142
6.2.1 Estimation algorithm principles . . . 142
6.2.2 Prediction for the uplink dynamic scheduler . . . 145
6.3 Evaluation . . . 146
6.3.1 Lab testbed . . . 146
6.3.2 Results . . . 147
6.4 Conclusion and Further studies . . . 153

**7 Towards an uplink scheduler combining latency, throughput and energy efficiency 155**

7.1 Motivation . . . 156
7.2 Radio resource allocation . . . 156
7.2.1 Scheduling algorithm . . . 156
7.2.2 Time-domain scheduling . . . 157
7.2.3 Frequency-domain scheduling . . . 158
7.3 Energy efficiency considerations . . . 159
7.4 Proposed uplink scheduling method using traffic prediction . . . 159
7.4.1 Adaptive periodicity of transmission opportunities . . . 160
7.4.2 Distribution of terminals over frames . . . 162
7.4.3 Link adaptation and fairness with Proportional-Fair . . . 162
7.5 Preliminary Evaluation . . . 163
7.5.1 Simulation setup . . . 163
7.5.2 Performance analysis . . . 165
7.6 Summary . . . 167

**Conclusion and Perspectives 169**

**Appendix 174**

**Bibliography** **201**

**List of Figures** **227**

**List of Listings** **231**

**List of Tables** **232**

**List of Symbols** **233**

**List of Accronyms** **234**

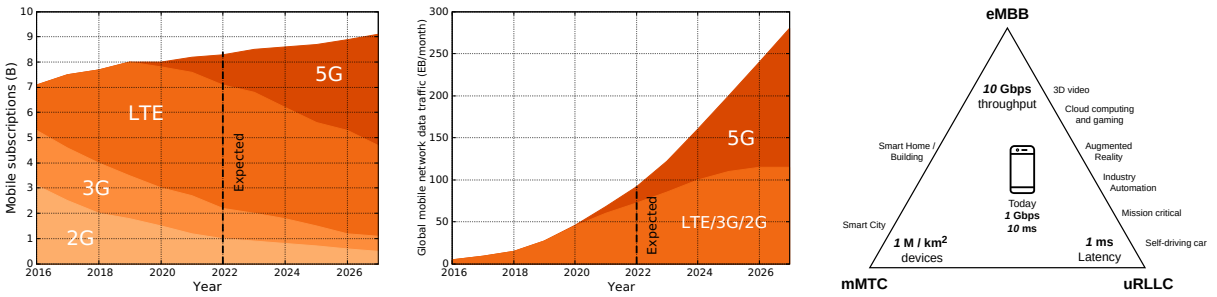
# INTRODUCTION

## Motivation

5G is the next generation cellular network technology, released in 2018 in replacement to LTE. According to [7] the number of 5G subscribers should increase to surpass LTE in 2026 (Figure 0.1a). In the same time, customers increasingly favor mobile access for their internet consumption [8]. Meanwhile, traffic intensive services has developed on mobile such as video streaming, gaming and conferencing. In consequence, The amount of traffic carried by mobile networks grows exponentially since 2016 as depicted in Fig 0.1b. Since the capacity of LTE networks seems saturated, especially in dense zone with high throughputs requirements at peak hours, as a most efficient radio access technology, 5G-New Radio (NR) [9] will make it possible to support this increase in communication needs [5]. By 2026, 5G should be used for the transmission of 130 EB per month worldwide in a total of 240 EB for all technologies. This will alleviate the traffic load for LTE networks but at the cost of a new infrastructure deployment.

5G-NR [1] is designed as the successor of the LTE Radio Access Technology (RAT) [10] by the Third Generation Partnership Project (3GPP) consortium. It was designed to be the global standard for the air interface of 5G networks. 5G-NR shares many similarities with LTE, in the manner to perform radio transmissions, however, the radio transmissions techniques have been enhanced to meet requirements defined by International Telecommunication Union (ITU) IMT-2020 for 5G [11]. These requirements are commonly represented in the form of the triangle in Figure 0.1c with 3 types of services targeted [12]. enhanced Mobile BroadBand (eMBB) for extreme throughputs towards 10 Gbps, massive Machine Type Communications (mMTC) for dense machine-to-machine communications and Ultra-Reliable Low-Latency Communications (URLLC) for extremely low latency towards 1 ms [10]. Nowadays, cellular networks are theoretically able to provides 1 Gbps with 10 ms latency. This is a complex challenge to meet required values knowing that there is a fundamental trade-off between throughput, reliability and latency in radio

5. "The ecological cost of 5G in 4 questions", Les Echos [Fr] (11/2020) `https://www.lesechos.fr/tech-medias/hightech/le-cout-ecologique-de-la-5g-en-4-questions-1228972`




(a) Evolution of the number of mobile subscriptions per technology

(b) Evolution of the traffic carried by mobile networks

(c) New services and capabilities of 5G.

Figure 0.1 – Emergence of 5G in mobile habits and new services (Source: Ericsson [7]).

transmissions [13].

That is particularly true for the latency which has been somewhat neglected in favor of throughput in the past. However, there has been a strong motivation to reduce latency in recent years with the emergence of services that require it. Self-driving automotive, industry automation, tactile internet and cloud gaming are usually given as examples of services which requires low latency and low jitter to be deployed in mobile networks [12, 14]. In fact, low latency is important in machine-to-machine communications timescale more than in machine-to-human interactions timescale.

Standards and recent researches propose a number of Radio Access Network (RAN) techniques to satisfy the needs of low latency in cellular networks [15], including shortened processing and better radio resource management.

Also, 5G redefines the end-to-end Quality of Service (QoS) model and network architecture to support specific services mentioned earlier. Besides, QoS architecture which prioritizes the access to radio resources of a user or a flow over another is not fully inline with the Internet philosophy of the "net neutrality". Because the Best Effort (BE) continues to be the main carrying mode of data volume in cellular networks (LTE, 5G), it is particularly interesting to focus on BE-enabled techniques to provide lower latency for the largest number of customers and for the largest number of innovative internet services.

The literature is rich in latency measurement campaigns in commercial cellular networks [16–18]. The *bufferbloat* phenomena [19] is shown as prevalent in cellular networks, exhibiting significant and persistent latency [20]. Bufferbloat is the consequence of a steadily filling large buffer at the path bottleneck. The RAN is an important bottleneck along the packet path. Large buffers are in use in the RAN to absorb important net-

work capacity variations and retransmissions that occur at the radio channel, favoring the constitution of a bufferbloat. The bufferbloat also impacts the efficiency of transport protocols, such as Transmission Control Protocol (TCP) and Quic (QUIC) (both represent more than 98% of internet traffic [21]) on achieving good usage of network capacity. The efficiency of TCP over cellular networks is often shown to be suboptimal with a radio capacity utilization of 60% [6], with a cost for operators who bought expensive radio spectrum. The question of downlink bufferbloat is well studied with many solutions using cross-layer techniques or active queue management in the RAN. In the same time, the bufferbloat [22] and more generally the latency of the uplink channel was less studied also because the downlink latency had a more important contribution to RAN Round-Time Trip (RTT) and because the majority of traffic was in the downlink. That will be no longer true with more and more latency-sensitive uplink services. 5G takes these evolutions into account with a more efficient uplink channel in terms of throughput (OFDMA, wider bandwidth) and latency (enhanced HARQ, optimized access method). These standardized methods are necessary but not sufficient to tackle uplink latency and bufferbloat for BE traffics.

The objective of this thesis is for reducing latency and jitter of the uplink channel in the 5G and LTE RANs for a majority of mobile customers. The fine-grain characterization of latency in the network is not an obvious problem, we also try to solve this problem in this thesis. These subjects are of great interest for a network operator, so this work is funded by Orange under a CIFRE contract.

# Thesis Contributions

This thesis is divided in three parts. In the first part, we have developed a latency-measurement tool called *LatSeq* to measure latencies of less than 1 ms in the RAN. In the second part, we identify and analyze the latency and jitter due to the uplink channel. We also give a practical example of the impact of this latency on the TCP transmissions efficiency and the radio resource utilization. In the third part, we propose a solution to tackle the identified uplink latency with a MAC module for the Base Station (BS) side in complement to the scheduler. The three main contributions of this thesis are summarized in the following.

**Contribution 1 : Improving understanding of RAN latency with LatSeq** We found a limited literature on in-radio access network latency at a packet level. After a review of open-source LTE and 5G projects, we were not able to find a lightweight tool to complete the analysis of these RAN latency at such level of detail. Thus, we propose a new deep internal latency inspection tool in the RAN called *LatSeq* implemented on OpenAirInterface LTE base station. We demonstrated both the low impact of measurements on running base station and the fine grain analysis of internal latency that could be made. We used outcomes for a better understanding of the latency dynamics per layer, per User Equipment (UE) and per packet, correlated with the radio context. Furthermore, we have worked with the OpenAirInterface (OAI) community to get our tool incorporated into the official repository of the open-source RAN to help the community to develop and debug new solutions for the 5G latency incoming challenges.

**Contribution 2 : Analyzing the contribution of uplink latency to RAN latency** Our evaluation of RAN latency in a lab testbed let us confirm the existence of a downlink bufferbloat in cellular networks as suggested by several previous studies. In the downlink, the latency is made of constant delay due to processing and transmission and a variable queueing latency in various buffers. The two mentioned delays as well as retransmission delay and handover latency in downlink are well studied in the literature. We show that even in a low-loaded cell and good radio conditions, there are a significant contribution of the uplink channel to the RAN RTT and jitter (i.e. variation of RTT). We explain this latency in an almost empty cell by the uplink radio error correction and recovery for a part but for a bigger part in good radio conditions, by the uplink radio resource allocation scheme and access method. We demonstrate by test experiments, field trial

and analytical expressions how grant-based uplink access is a source of a non-negligible delay and jitter.

**Contribution 3 : Addressing the uplink latency and jitter problem with Enhanced-BSR, an evolution of buffer estimation algorithm for MAC layer** One of the cause of mentioned uplink access latency, lies in a bad knowledge of the mobile terminal needs in transmission by the base station. The proposed solution *enhanced-BSR* opts for an estimation method of the UE needs in transmission. We have proposed a new buffer status estimation algorithm for the base station's MAC uplink scheduler which utilizes "Scheduling Request" (SR) and "Buffer Status Reports" (BSR) and transport block utilization. The implementation is 3GPP 5G and LTE standards compliant and was made in the OAI base station. We demonstrated with experiments on selected types of traffic (TCP transfer and uplink traffics) that the new estimation could accurately predict UE transmission buffer length. The outcome is a more efficient radio resource allocation by the BS dynamic scheduler for access latency and jitter resulting in an important reduction of the uplink jitter in the presented radio context and mobile traffics.

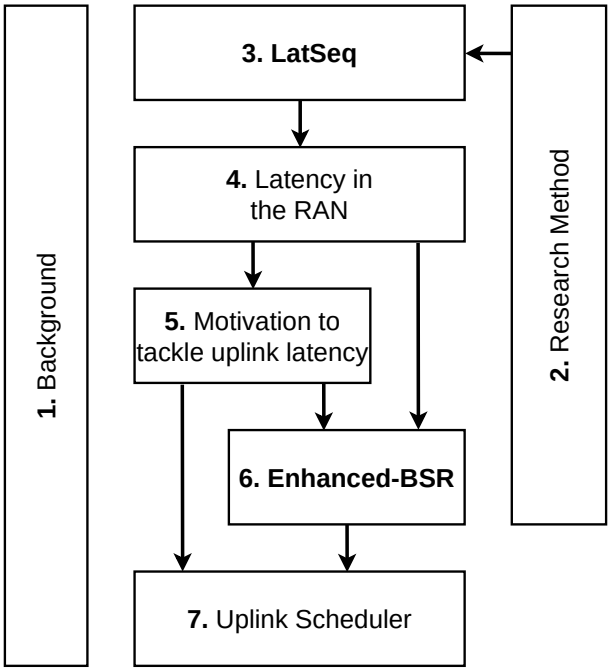



Figure 0.2 – Thesis chapters outline

# Dissertation outline and organization

The rest of the thesis is organized in chapters as follows. The links between chapters are illustrated in Figure 0.2. Elements of understanding concerning cellular networks and internet protocols, followed by a review of latency in the access network and their mitigation are given in Chapter 1. The testbed and research methods are detailed in Chapter 2. Chapter 3 presents the LatSeq tool developed for internal latency analysis which completes the testbed. A detailed analysis of latency in the RAN is provided in Chapter 4 with a focus on the uplink channel. Furthermore, Chapter 5 motivates the need to reduce jitter and uplink latency to improve TCP efficiency with ultimately better utilization of radio resources and to reduce end-to-end latency. Our proposed method to reduce uplink latency jitter with a predictive estimation model of the transmission needs, enhanced-BSR, is explained in Chapter 6. Following that, a preliminary work explores perspective opened by enhanced-BSR for the uplink scheduling in Chapter 7. The manuscript ends with a general conclusion and perspectives of this thesis work.

Chapter 1

# Background

**Abstract** : *This chapter provides the elements of understanding for the rest of the thesis. The first section is an overview of 5G and LTE cellular systems, with a focus on the Radio Access Network (RAN) and the radio resource allocation. The second section presents the internet protocol suits with an introduction to Transport Control Protocol (TCP), congestion control and Quic protocol. Finally, the last section is dedicated to the latencies in the networks and their mitigation techniques. After latency definitions, to issue of bufferfloat is exposed followed by a short review of latency mitigation techniques in 5G.*

## Contents

**1.1 5G and Cellular Networks . . . . . 8**
1.1.1 Cellular network architecture . . . . . 9
1.1.2 Radio Access Network . . . . . 12
1.1.3 Radio resource allocation . . . . . 17
**1.2 Internet Protocol Suite . . . . . 26**
1.2.1 Transmission Control Protocol (TCP) . . . . . 26
1.2.2 Congestion control . . . . . 26
1.2.3 Quic . . . . . 27
**1.3 Latency in Networks and Mitigation . . . . . 29**
1.3.1 Definition of latency . . . . . 29
1.3.2 Bufferbloat . . . . . 32
1.3.3 Latency Mitigation in 5G . . . . . 36

## 1.1 5G and Cellular Networks

A new generation of Radio Access Technology (RAT) is launched every decade since 1980. 3G in 1990, 3.5G HSDPA in 2002, 4G in 2009. Each generation bringing new capabilities to mobile networks for instance, web-browsing with 3G and video streaming with 4G. In 2010's Long Term Evolution (LTE)-Advanced and Advanced pro was launched with incremental improvements of LTE. The Fifth Generation (5G) RAT is introduced in 2019 to come after LTE technologies. After a long phase of exploration and standardization Third Generation Partnership Project (3GPP) issued Release 15 [1] describing New Radio (NR), the new 5G RAT [2] [9] in accordance with requirements edicted by International Telecommunication Union (ITU). Release 15 describes initial 5G RAT i.e. how Radio Access Network (RAN) works and how to connect it to a mobile core. Since then, Release 16 [3] (2020) [23] and Release 17 [4] have brought improvements in many areas such as signalling, architecture and services provided.

5G Non-StandAlone mode (NSA) (using an LTE mobile core) is already deployed for commercial uses with more than 600 million subscribers during the first quarter of 2022 and expected to be 1 billion by then end of this year worldwide [7]. Figure 0.1a depicts the expected number of subscribers for the different RAT until 2027 and shows a rapid growth of the 5G usage in the next few years, mainly in replacement of 3G and 2G. It comes with an explosion of usage and data consumption as illustrated in Fig. 0.1b permitted by new 5G capabilities. The popularity of the video streaming everywhere and the development of new data intensive services such as mobile cloud gaming, Virtual Reality (VR) and remote work saturate a number of cells in dense area, reaching capabilities of LTE. 5G must be able to support the need for capacity in 2030 thanks to a redesigned radio layer and mobile network architecture. When theoretically the maximum peak datarate in LTE is 1 Gbps, 5G offers a maximum of 10 Gbps in downlink and uplink. In this task, an increase of spectral efficiency, number of cells and of spectrum bandwidth are needed, implying a number of challenges in radio layer.

A fundamental movement in past years has been the convergence between the various wireless networks and fixed networks. LTE/5G and Wifi uses the same modulation scheme, home fiber and 5G seek to provide 1 Gbps symmetric to customer. In the same

1. `https://www.3gpp.org/release-15`
2. ShareTechnotes : `http://www.sharetechnote.com/html/5G/`
3. TR 21.916 : `https://www.3gpp.org/release-16`
4. TR 21.917 : `https://www.3gpp.org/release-17`

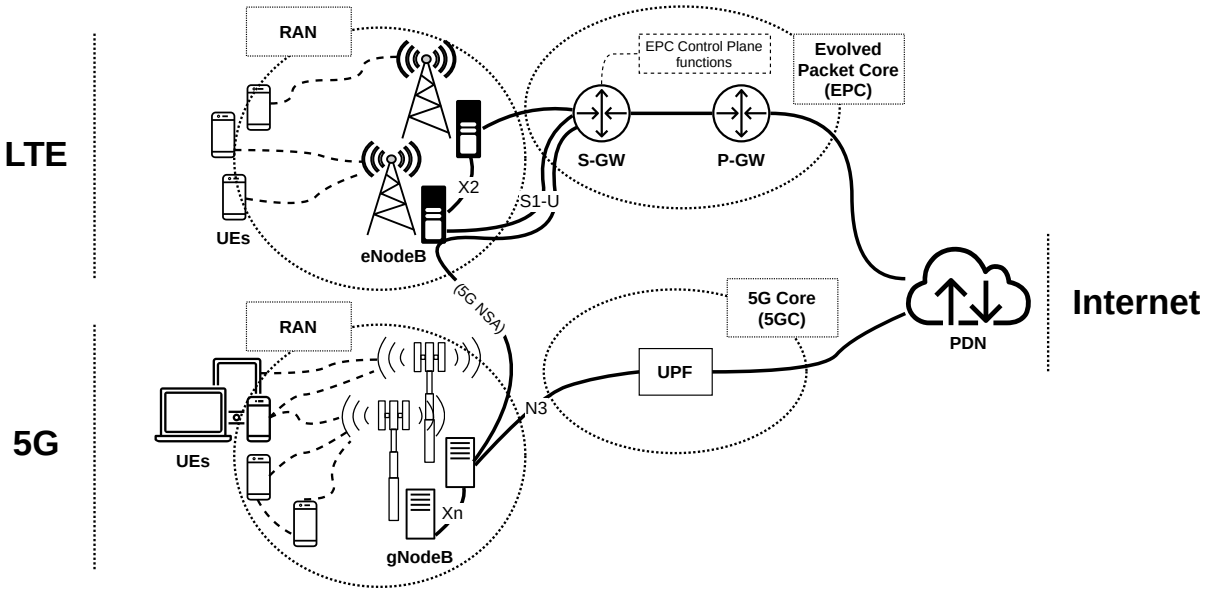



Figure 1.1 – LTE and 5G user-plane networks architecture with a focus on the user-plane.

idea, it is desired that the latencies of fixed and mobile networks converge. ITU IMT-2020 defines an objective of 1 ms RAN Round-Time Trip (RTT) for 5G (see Fig. 0.1c), when the minimum latency achievable in LTE is about 20 ms. Latency is probably the most difficult challenge to solve in 5G because there are at the same time fundamental limits related to the radio channel [13] and questions of architecture of the mobile network.

Latency mitigation techniques for 5G will be discussed in Section 1.3.3. In this section, we first give an insight of current and upcoming cellular network architectures. After that, an overview of cellular Radio Access Network (RAN) protocol stack §1.1.2.1 and physical radio channel will be presented §1.1.2.2. Radio resource allocation schemes will be detailed in §1.1.3 with a special focus on the uplink channel.

### 1.1.1 Cellular network architecture

The cellular network connects mobile terminals (i.e. User Equipments (UEs)) to the Internet Protocol (IP) network (i.e Packet Data Network (PDN)). Both LTE and 5G cellular networks consists in 2 parts, the RAN (i.e. Evolved Universal Terrestrial Radio Access (E-UTRA) for LTE and Next Generation RAN (NG-RAN) for 5G) and the Mobile Network Core (MNC) (i.e. Evolved Packet Core (EPC) for LTE and 5G Core (5GC) for 5G) [24]-III. An overview of LTE and 5G cellular network architectures are depicted in Figure 1.1. On the left are the mobile terminals and on the right the IP network. UEs

are connected to Base Station (BS), called Evolved NodeB (eNB) in LTE and gNodeB (gNB) in 5G through the radio air interface in the RAN. BSs are then connected to MNC through backhaul network (in practice, a high capacity optical fiber of hundred kilometers connecting the base station to the datacenter where the core is located). MNC is the edge interface with PDN.

**Radio Access Network** The radio access network consists in cells served by one or more BSs and UEs. Area covered by a cell depends on the UE density, geography of the area, antenna power and frequency used. BS provides radio functionalities (Radio-Frequency (RF), power amplification, beam forming, etc.), baseband processing (data coding, modulation, radio resource mapping, etc.) and radio link layer-2 (Medium Access Control (MAC), Radio Link Control (RLC), Packet Data Convergence Protocol (PDCP), Radio Resource Control (RRC) protocols). BSs are connected together with X2 interface (Xn in 5G) to handle mobility of a user between cells with handover procedure (i.e. performing user context and data migration from one cell to another). BSs cooperate to offer the best throughput and energy efficiency (e.g. balancing connected UEs between BSs to share traffic load). Also, 5G is designed to operate with the LTE network [25]. It is necessary in 5G NSA since the control place is provided by eNB and EPC. In the future, interworking between the 4G and 5G network will improve connectivity while distributing the traffic load.

UEs in the cell are either connected and active or sleeping and inactive. After an initial access procedure, the UE receives a Radio Network Temporary Identifier (RNTI) that is an identifier for this UE in this cell for the duration of the session. RNTI is transported by the MAC layer header and serves to route and decode radio messages. Also, a Data Radio Bearer (DRB) for user traffic and a Signal Radio Bearer (SRB) for the control-plane traffic are instantiated for the UE. Radio bearers consist in practice on virtual circuits identified with a Logical Channel IDentifier (LCID) associated to layer software entities.

More DRBs can be instantiated in addition to the default bearer to set up virtual circuits for specific traffics e.g. voice carrying (see Voice over LTE (VoLTE) bearer in [26]-11), latency-sensitive slices [27][28]-13.7. A QoS Class Identifier (QCI) (or 5 QoS Identifier (5QI) [29]) is associated to the bearer which specifies the level of Quality of Service (QoS) requirements in terms of delay, datarate and priority [5] [6].

5. LTE Channel Quality Indication (CQI) table : `https://www.sharetechnote.com/html/Handbook_LTE_QCI.html`

6. 5G 5QI table : `https://www.sharetechnote.com/html/5G/5G_5QI.html`

**Mobile Network Core** with a focus on user-plane. Radio bearers end in the core network. The architecture of the LTE EPC is depicted at the middle top on Fig. 1.1. Control-plane and user-plane traffics are routed by the Serving Gateway (SGW), to the network functions and Packet Gateway (PGW) respectively. User plane traffic is then routed to the PGW that is the IP edge for EPC. PGW is a kind of proxy that allocates IP addresses to users and connect them to the PDN. The architecture of 5GC has been redesigned to be more flexible and more open with a "cloud-native" strategy. The principle is the same, User Plane Function (UPF) plays the role of PGW and ensures to connectivity of user traffic to Internet and Access and Mobility Management Function (AMF) plays the role of SGW in terms of mobility and access management. The new 5G core architecture is designed to manage end-to-end flows in the mobile network in accordance with the new QoS framework.

### 1.1.2 Radio Access Network

RAN ensures the connectivity of mobile devices to the mobile network. At each end of the radio channel there are devices and a base station. 3GPP Technical Specifications (TS) 36.300 [30] and 38.300 [31] define respectively for LTE and 5G radio protocol architecture to permit interoperable communication between UE and BS through the physical radio interface. The protocol stack is discussed in Figure 1.2 and in the following. Thereafter, we make a brief presentation of the physical radio layer (§1.1.2.2).

#### 1.1.2.1 LTE and 5G protocol stack

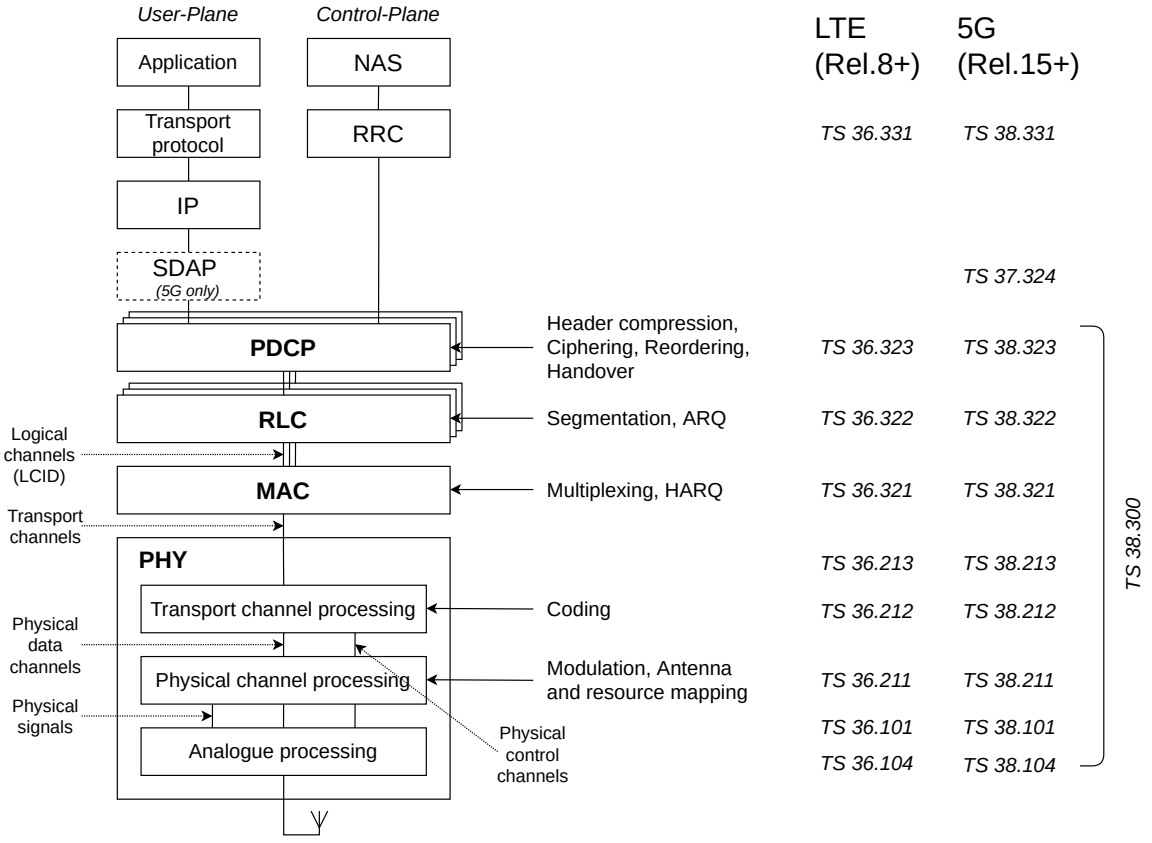



Figure 1.2 – LTE and 5G protocol stack and 3GPP Technical Specifications.

Figure 1.2 summarizes protocols in use in the RAN. The same stack is implemented in both BS and UE modem. OSI layer 2 includes Packet Data Convergence Protocol (PDCP), Radio Link Control (RLC) and Medium Access Control (MAC) protocols. "Application", "Transport protocol" and "IP" are given for information to represent user plane

while RRC [32, 33] represents the radio control plane. User plane and Control plane traffics shares the same protocol entities after the PDCP. However, user and control traffic are isolated in 2 different bearers, respectively in DRB and SRB. Also, as can be seen in the figure, 5G has taken over the protocol architecture of LTE with great similarities between these protocols. Thus, many remarks on the radio layer 2 in LTE remain true in 5G. The 5G Service Data Adaptation Protocol (SDAP) entity is put aside here because it will not be implemented during this thesis. SDAP is related to the new 5G QoS framework with QoS flows and not directly for radio transmission operations.

**Packet Data Convergence Protocol (PDCP)** layer [34, 35] is responsible for security (ciphering and integrity protection), routing (mapping of radio bearers to RLC entities, routing to cell group in dual connectivity, inter-cell handover handling) and header compression (compression of IP header with Robust Header Compression (ROHC) to reduce overhead). PDCP also implements packet retransmission and in-sequence delivery functionalities to ensure reliability of the radio link in complement to RLC and MAC layers. Latencies associated to PDCP layer are:

— Processing delay for data encryption
— In-sequence delivery when enabled could cause Head-of-Line (HoL) blocking
— The handover procedure with the transfer of the PDCP entity buffer's contents from one BS to another, with also a risk of a packets discarding.

**Radio Link Control (RLC)** protocol [36, 37] prepares packets from PDCP entity for the MAC layer (segmentation [7] and sequence numbering). Packets awaiting to be transmitting are stored inside the transmission buffer. RLC entity identified with a LCID belongs to one radio bearer. According to the traffic carried by the logical channel, there are 3 entity modes:

— Transparent Mode (TM) where the RLC entity is totally transparent without segmentation and retransmission. It is used for control-plane broadcast where the information should reach multiple users.
— Unacknowledged Mode (UM) supports segmentation but no retransmissions, used when error-free delivery is not required. It is recommended in the literature for VoLTE but not for internet services (despite TCP protocol could handle a low error rate like one at the output of the MAC layer after Hybrid-ARQ (HARQ) process).

7. and concatenation in LTE

— Acknowledged Mode (AM) is the default mode for the user plane DRB. RLC Automatic Repeat reQuest (ARQ) procedure is implemented for error detection and recovery handling. An ARQ feedback loop for acknowledgment and retransmission buffer are needed to ensure retransmission process, adding latency, control signalling and RAN configurations. It makes AM more robust and reliable, but this induces delay from the control overhead as opposed to cutting losses.

Latency associated to the RLC layer are:

— Buffering latency since the RLC entity stores packets from IP layer for radio layer. It is very dependent on the buffer size, except that we know that buffers in mobile networks are big (several megabytes), to accommodate variations in capacity and retransmission needs.

— Retransmission latency for AM entity and the ARQ feedback delay to detect and recover lower layers losses. For that reason, the UM should be preferred for latency-sensitive flows such as video streaming, and voice-over-ip.

**Medium Access Control (MAC)** [32, 33] entity is common to all bearers in a BS or a UE and multiplexes logical channels. Because of that, BS's MAC entity is responsible for scheduling of users and bearers for both uplink and downlink. UE's MAC entity is in charge of the prioritization of the instantiated bearers. Scheduling and radio resource allocation schemes will be discussed in detail Sec. 1.1.3. MAC encodes Transport Blocks (TBs) to be transmitted through radio physical layer (conversely decode them) by concatenating RLC data segments from multiple logical channels, MAC header and eventually padding to fit with the scheduled Transport Block Size (TBS).

The MAC Hybrid-ARQ (HARQ) process, up to 8 in parallel, handles most of the radio link errors (10% error at the first MAC transmission [13]-Table I, less than 1% at the RLC layer and less than 0.1% at the PDCP layer) with recovery and retransmission of TB. HARQ is the combination of an Automatic Repeat reQuest (ARQ) and a Forward Error Correction (FEC). FEC refers to a more general method which tries to detect and correct errors in a transmission. When an unrecoverable error appears, the decoding HARQ entity sends a Not-ACK (NACK) to the transmitting HARQ counterpart for a retransmission. 5G modified HARQ method to be more efficient in error recovery with Incremental Redundancy (IR) technique and more latency efficient with a shortened ACK feedback loop.

Finally, MAC layer manages number of radio-related network functions such as random

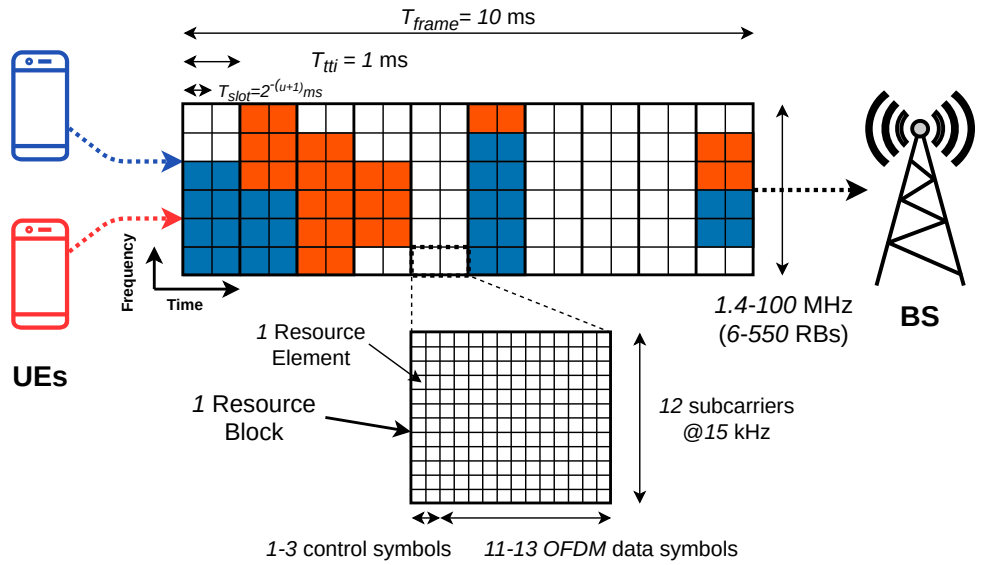



Figure 1.3 – The OFDMA uplink radio resource grid shared between 2 UEs (blue and orange). *The radio resources allocated to physical control signalling are not represented.*

access control (for initial access) and Link Adaptation (LA) mechanism. The role of the LA is to adapt Adaptive Modulation and Coding (AMC) used in Physical Layer (PHY) using CQI. TBS is also determined by the MAC according to AMC.

Latency related to the MAC layer are:

— Processing delay for multiplex and de-multiplex transport blocks

— Scheduling latency depending on the current cell load, position of the user and the scheduling strategy

— DL and UL signalling procedures to access radio channel

— HARQ feedback loop and retransmission delays of about 8 ms in LTE.

For further information, an in-depth representation of the protocol stack is given in Appendix C.C.4.

#### 1.1.2.2 Radio physical layer

There are fundamental trade-offs between capacity, coverage, latency, reliability, and spectral efficiency in cellular networks [13]. Network capacity is the theoretical maximum amount of data that the link can support [38]. The objective of the radio link layer is to achieve the maximum achievable network capacity.

To tend towards the Shannon capacity, many improvements have been introduced with the NR technology [1] from modulation and coding scheme to Multi-User Multiple-

Input Multiple-Output (MIMO) (see II.C of [10]-II.C). In reality, main differences between LTE and 5G RATs are related to physical layer to increase capacity and to support low-latency [24]-Table II. However, 5G like LTE utilizes Orthogonal Frequency-Division Multiple (OFDM) as transmission scheme, quadrature-amplitude for modulation and support Frequency-Division Duplexing (FDD) and Time-Division Duplexing (TDD) duplexing modes.

In Time-Division Duplexing (TDD), uplink and downlink multiplexing is made in time with an alternation of the uplink and downlink transmissions. Frequency-Division Duplexing (FDD) is very common in commercial network, uplink and downlink radio transmissions are multiplexed in frequency with one band for the uplink and one band for the downlink. It is robust and permit a continuous transmission of data in both directions in the same time. Band 7 for example reserves band $2500 - 2570$ MHz for uplink and $2620 - 2690$ MHz for downlink.

After that, the radio channel is organized as a *time-frequency resource grid* composed of Physical Resource Blocks (PRBs) as depicted in Figure 1.3. Resource grid is shared between all users in the cell. The figure depicts a Single-Input Single-Output (SISO) situation, when for a MIMO system, there is one different resource grid for each antenna.

In the time domain, transmissions are organized by *frame* of 10 ms, *subframe* of 1 ms and 2 slots per sub-frame. At most one TB of dynamic size is transmitted during a *Transmission Time Interval (TTI)* of 1 ms through the air interface.

PRBs are divided in frequency-domain into 12 subcarriers and in time into 14 OFDM symbols carrying encoded and modulated data. The PRB is allocated to one user, but the bandwidth composed of several PRBs could be allocated to several users in the same time with the Orthogonal Frequency-Division Multiple Access (OFDMA) model. OFDMA allows flexible and dynamic sharing of the bandwidth between UEs. It is used for the downlink channel in both LTE and 5G and also for the uplink in 5G. The access method for the uplink in LTE is slightly different since it uses the Single-Carrier FDMA (SC-FDMA) and is not represented on the figure.

Data are encoded into a TB with a size determined by the number of PRB and the Modulation and Coding Scheme (MCS). In particular, the MCS is associated to a Quadrature-Amplitude (QAM) scheme [39]-Table-7.1.7.1-1 and [40]-Table-5.1.3.1-1,2, according to the channel quality. The variation of the modulation order and coding rate gives the spectral efficiency achievable for a symbol in bits per second per Hz, from 0.15 to 5.55 bit/s/Hz in LTE and 0.23 to 7.44 bit/s/Hz in 5G. Consequently, an LTE PRB could

carry from 3 bytes [8] to 124 bytes of data. The bandwidth goes from 1.4 MHz to 20 MHz and 100 MHz in 5G with carrier-aggregation and MIMO rank, increasing proportionally the available radio resources and achievable throughput [41].

Some OFDM symbols of a resource block are used for control channels [9], for instance Primary Synchronization Channel (PSCH), Secondary Synchronization Channel (SSCH), Physical Downlink/Uplink Control Channel (PXCCH) or Physical HARQ Indicator Channel (PHICH). Usually, it is admitted to have 14% of control overhead on radio capacity.

The Physical Downlink Control Channel (PDCCH) carries downlink scheduling assignments, information necessary for the mobile to be able to receive, demodulate and decode downlink user channel data and transport control elements to permit uplink transmissions. The payload carried on the Physical Downlink Control Channel (PDCCH) is the Downlink Control Information (DCI) encoded with the RNTI of the device. The Downlink Control Information (DCI) message is of different format, format 0 carries information related to the uplink scheduling grants and format 1 information related to the downlink assignments.

To learn more about the LTE physical layer, an excellent tutorial is proposed in [42] which presents physical layer from theory to practice with a MATLAB implementation. In addition, the research work presenting an implementation of an end-to-end 5G NR simulation [43] propose a tutorial on the physical and MAC layer in 5G.

### 1.1.3 Radio resource allocation

In the previous Sec. 1.1.2.2, we gave an overview of the physical radio layer. Especially, we detailed the radio time-frequency resource grid composed of PRBs shared between UEs. In this section, we discuss the radio resource allocation schemes with a focus on the *uplink radio resource grid* (Fig.1.3) at the RAN-level.

#### 1.1.3.1 Allocation schemes

RAN is essentially a scheduled system where time-frequency radio resources are assigned to mobile devices. Scheduling for the uplink and downlink channels follows roughly same principles except for the fundamental asymmetry that exists between the two. The transmission power and data packets are distributed among UEs when in the downlink

8. $15 \cdot 10^3 \times 12 \times 0.15/1000/8$

9. A graphical generation of an LTE radio resource grid in `https://www.sqimway.com/lte_resource_grid.html`

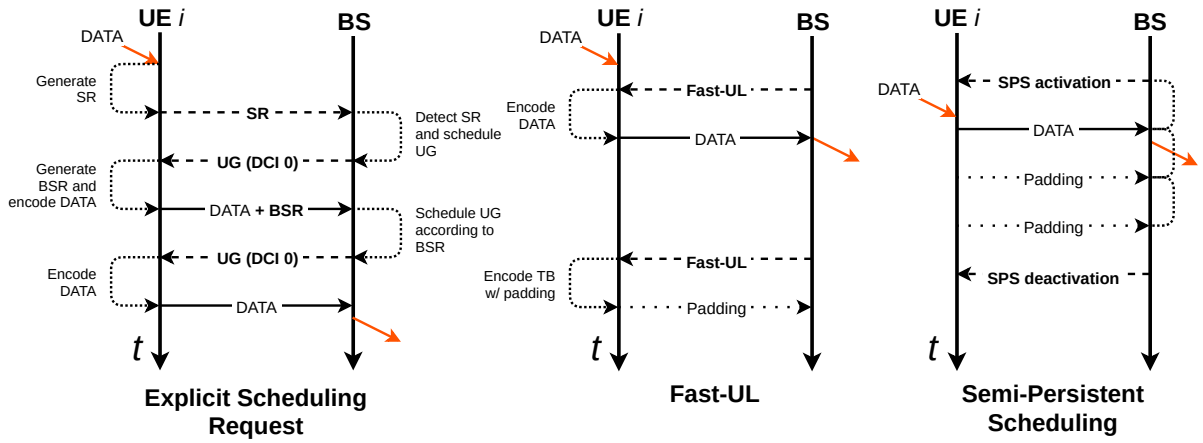



Figure 1.4 – Uplink radio resource allocation schemes. *Explicit Scheduling Request is the by-default used allocation scheme.*

power resource and packets are centralized within the BS. Moreover, BS is responsible for the scheduling for the whole cell.

**Downlink dynamic scheduling** In the downlink, the default radio resource allocation is made dynamically by the BS MAC scheduler. For each TTI the scheduler shares PRBs between DRBs with Protocol Data Unit (PDU) in the RLC transmission buffer. The quantity of PRBs (i.e. the TBS) allocated to a DRB is decided according to a scheduling strategy, parameters (QoS requirements), and the network status (buffer occupancy, channel quality). The UE is informed of the reception of a transport block in a TTI with parameters to decode it from the DCI in the front of the TTI.

**Uplink dynamic scheduling** In the uplink, the default radio resource allocation is also made dynamically by the BS MAC scheduler. But at the difference of the downlink, the user have to request access to the channel with a Scheduling Request (SR) (§1.1.3.2), that we call *explicit scheduling request access*. In response the BS MAC scheduler (§1.1.3.2) grants user with few radio resources with an Uplink Grant or grant of uplink capacity (UG), that is called grant-based access. SR is completed by the Buffer Status Report (BSR) (§1.1.3.2). The procedure is depicted in Fig. 1.4 in the left part. This grant-based access is primarily used for uplink radio resource allocation because it solves the contention problem with a dynamic allocation that comes with a shared radio resource grid. It is particularly adapted for rapidly changing radio channel capacity and traffic

datarate with many served users. We will discuss more about uplink dynamic scheduling in the further section § 1.1.3.2.

**Fast Uplink access** or pre-scheduling or configured-grant is a grant-based technique where the BS transmits a UG without the prior reception of a SR. The procedure is depicted in Fig. 1.4-"Fast-UL". The BS typically predicts when the UE should need access to the uplink channel to transmit new data. A small share of the bandwidth is commonly allocated regularly (few resource blocks). Thus, a good Signal-to-Noise Ratio (SNR) is necessary to transmit a IP packet in one transmission, limiting the technique to the users with a good radio conditions. Fast Uplink Grant (Fast-UL) is considered as an important low-latency enabler in RAN on the condition of a good estimation of data packet arrival by the BS (e.g. [44] studied the soft-reservation for latency-sensitive teleoperation in mobile networks). UE could transmit padding when the transmission buffer is empty, but it is not mandating as with Semi-Persistent Scheduling (SPS) (further complexity partly solved by using HARQ NACK and soft-combining).

However, resources are allocated to one user with the risk of a false estimate which would lead to radio resource wastage impacting the rest of the cell performance. The explicit SR method is more conservative than the Fast-UL concerning the radio resource allocation by granting only the resources requested by the UE.

**Semi-Persistent Scheduling** is a grant-free semi-static allocation technique by which a UE is provided with reserved radio resource during a period of time at a configured periodicity. The periodicity and the reserved resource blocks are network-configured with an RRC message prior to the activation of the SPS. L1/L2 RRC control activate and deactivation the SPS channel. Figures 1.4-"SPS" illustrates a data transmission when SPS is enabled with a fixed periodicity for transmission. TB padding occurs when there are no data left in the transmission buffer. SPS was standardized primarily for Voice over IP [45, 46] but it is also as a solution for latency and jitter sensitive services with a deterministic traffic pattern. Dynamic scheduling is able to take into account radio variation of traffic generation and radio channel capacity while semi-static scheduling needs prior transmission parameters.

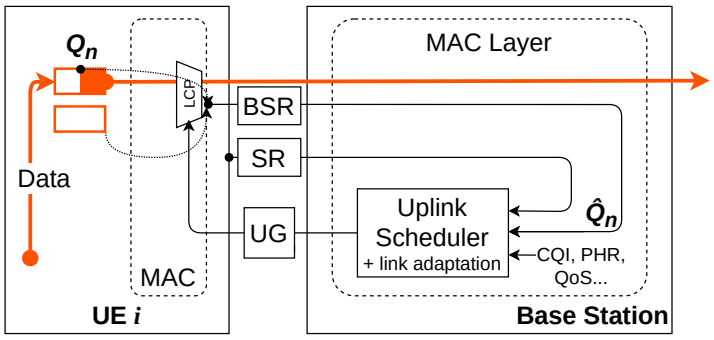



Figure 1.5 – Uplink grant-based dynamic scheduling

#### 1.1.3.2 Uplink dynamic scheduling

The uplink dynamic grant-based scheduling scheme is composed of 5 elements illustrated in Figure 1.5:

— The uplink scheduler (BS MAC control)
— Uplink Grant or grant of uplink capacity (UG) (physical downlink control channel) or grant of uplink capacity
— MAC multiplexing and Logical Channel prioritization (LCP) (UE MAC layer)
— The Scheduling Request (SR) signalling (physical uplink control channel)
— The Buffer Status Report (BSR) signalling (uplink MAC Control Element (MAC CE))

3GPP standards describe procedures and mechanisms associated to dynamic scheduling in [31–33, 39, 47–49].

**MAC scheduler** LTE and 5G standards concerning uplink scheduling look very similar except 5G introduces a new QoS model and more scheduling flexibilies at the physical layer [50, 51]. The 3GPP standard [31]-§10.3 defines the dynamic radio resource allocation by the BS and specifies that UEs always monitors the PDCCHs in order to find possible grants for uplink transmission when its downlink reception is enabled (see Discontinuous Reception (DRX) mechanism).

The section §10.1 of the technical specification [31] describes the basic scheduler operations which are:

— Taking into account the explicit scheduling request, UE buffer status and the QoS requirements
— May assign resources taking account the radio conditions at the UE

— Assign radio resources in a unit of a slot, with a minimal quantum of radio PRB

The operation definition does not standardize how to perform the scheduling according to Scheduling Request (SR)/Buffer Status Report (BSR) or QoS requirements. Scheduling algorithm is a hot topic of research for RAN performances [52–59] Especially, 5G scheduler comes with a larger number of options due to physical layer flexibilities [60] and QoS framework [1]-6.2. The MAC technical specification document [33]-§5.4 defines the Uplink Shared Channel (UL-SCH) data transfer. In particular, section §5.4.4 defines the procedure for UE to request UL-SCH resources for new transmission with a Scheduling Request (SR).

Scheduler is *channel-dependent* since 3G where transmissions to a device take place when the radio channel conditions are favorable and exploit channel variations. BS MAC scheduler is also in charge of the Link Adaptation (LA) [61], including Power Control (PC). PC has an important role since it determines the transmission energy that should use UE to transmit with a target Block Error Rate (BLER) of maximum 10% giving a MCS. The transmission power is an important source of energy consumption by mobile [62], and an important performance criteria for mobile users and it is reduced with a too high transmission power while a too low transmission power generates radio link errors and retransmissions.

The outcome of the uplink scheduler is the radio resource assignment indicated by the UG.

**Uplink Grant** indicates in the downlink DCI time-frequency radio resource allocation to the UEs. The information in the DCI formats 0-0 or 0-1 precises the UE's RNTI, information about resource (frequency-domain allocation, time-domain allocation), about transport block (MCS, redundancy version, HARQ process number), about antenna (antenna ports, MIMO scheme) and about uplink power control. An UG is necessary to transmit data over a PRB and is the final outcome of the scheduling process that occurs every TTI.

**Scheduling Request** A SR is triggered when a new data arrives in the UE transmission buffer. Thus, the UE will send SR (the number of pending SR is limited by a configuration) until it has new transmission. Pending SR sent by UE are reset when a UG is received. SR *periodicity* configuration [63] consists of a set of Physical Uplink Control Channel (PUCCH) resources on which SR flag could be transmitted. The quantity of information

| | **Scheduling Request** | **Buffer Status Report** |
|---|---|---|
| **Layer** | PHY | MAC |
| **Channel** | PUCCH | Uplink MAC CE |
| **Size** | A part of an OFDM symbol | 1 to 4 TB bytes |
| **Information carried** | Request access | RLC tx buffer size |
| **Granularity** | 1 bit | 6 bits (LTE) |
| **Tx opportunity** | every 1-80 subframes (typically 10) | at least every 1-2560 ms (typically 32) |

Table 1.1 – SR and Buffer Status Report (BSR) comparison.

carried by the scheduling request is limited (basically one bit) but could be transmitted regularly, typically, every 10 ms.

**Buffer Status Reporting** *TS 3GPP 38.321* [33] section §5.4.5 details the procedure to provide the serving base station with information about uplink data volume in the RLC entity, i.e. Buffer Status Report (BSR). The data volume calculation is detailed in the [37]-§5.5. The BSR MAC CE (in Fig. 1.6a) reports the buffer occupancy associated to a logical channel group (≃DRB) with an index of 6-bits length. While the buffer status reporting is made per DRB group, the UG is given to one UE. This scheme has been discussed at the 3GPP in 2006 [64, 65]. Pradap et al. in [66] reviews the different uplink buffer status reporting schemes considered by the 3GPP working group.

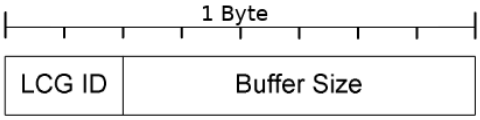
The reporting scheme is one trade-off. The other one is the BSR index table:

— In the width of the index range. An increase of this range implies an increase of the MAC CE size and then the MAC control overhead.

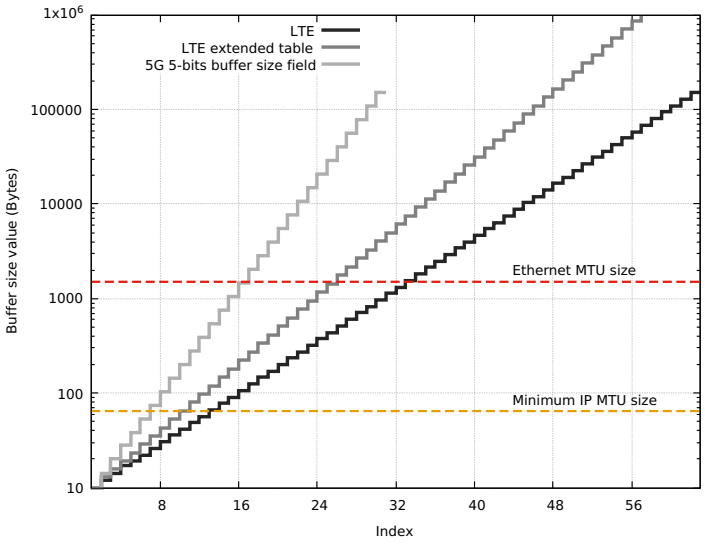
— In the maximum value reported by the range. A too little value limits the maximum uplink throughput reporting and a too large value decrease the accuracy of the reported value.

— The distribution function that takes a buffer size in input and returns an index. This function could be defined to split the range equality, but the standard choose to give importance to lower values. For high index values, the accuracy of reported bytes decreases.

The BSR reported values of buffer occupancy according to the LTE's BSR table,




(a) BSR MAC control element format. *LCG ID : Logical Channel Group Identifier; Buffer Size : BSR table index*




(b) Distribution of reported value by BSR per index in LTE and 5G.

Figure 1.6 – 3GPP Buffer Status Reporting.

LTE's extended BSR table and 5G's 5-bits BSR table are plotted in Figure 1.6b. The y-axis uses a log scale, showing that between 2 index in higher value, the accuracy of the reported buffer status decreases exponentially. A Maximum Transmission Unit (MTU)-length packet is reported with the index 33 in LTE, 26 with the extended table and 17 with the 5-bits 5G's BSR table. 5G standard defines 2 BSR formats, one with an index coded on 5 bits (32 possible values) and one with 8 bits (256 possible values). The 5 bits table encodes buffer occupancy from 0 to 150 kB as in with LTE table with a reduced accuracy on higher values.

Three types of event trigger a BSR:

— When a new UL data is available in logical channel with a higher priority : *Regular BSR*

— To fill padding bits when a UG allocates more bytes than it is necessary to empty the buffer : *Padding BSR*

— At the expiration of a timer, the `periodicBSR-Timer` : *Periodic BSR*

When transmission is running, the regular BSR should not be triggered in our experiment since we are using only the default non-QoS bearer. However, the transmission of a regular BSR is the result of an internal modem's process that it is neither network-configured nor standardized. The events that trigger BSR on our setup are the padding of a TB and the expiration of the `periodicBSR-Timer` when a TB is generated.

When the reported BSR value in LTE is 63, the base station estimates there are at least $Q_{\text{BSR}}^{\text{max}} = 150$ kB in the transmission buffer even if the UE transmission buffer is larger. The upper bound of BSR index $Q_{\text{BSR}}^{\text{max}}$ table and the BSR periodicity $T_{\text{BSR}}$ limit the estimated uplink packet generation datarate of the UE such as:

$$U_{\text{reported}}^{\text{max}} = Q_{\text{BSR}}^{\text{max}} / T_{\text{BSR}} \tag{1.1}$$

The value of periodicity is network-configured and usually set to 16, 32 or 64 ms [10] For a BSR reported every 64 ms, it implies $D_{\text{reported}}^{\text{max}} = 18.75$ Mbps. In the other side, a reported buffer size of 150 kB will be cleared in 11 or 400 TTIs in LTE (for a 20 MHz bandwidth) according to the MCS. Thus, the periodicity of BSR should be adapted according to the cell capacity and cell bandwidth to efficiently report the UE uplink traffic generation datarate. However, it is not the case in operated networks with fixed default values.

#### 1.1.3.3 Summary

In summary, The default radio resource allocation in the radio access network is dynamic and the result of the BS MAC scheduler. The UG informs devices for an uplink transmission opportunity. We presented 3 uplink access methods in Fig. 1.4. SPS and Fast-UL efficiently reduce uplink access latency. A latency analysis of each allocation scheme is provided in [55] and in [67]. However, SPS suffers from a lack of adaptation to condition changes and control overheads and Fast-UL is subject to radio resource wastage when UGs are not adapted to the transmission needs.

The explicit SR method is based on scheduling request and buffer status reporting to adapt UGs to transmission needs. The typical time-sequence radio resource allocation committed in the standard is illustrated in Figure 1.4-"Explicit Scheduling Request". A new data arrives at the UE transmission buffer that triggers a Scheduling Request (SR) with a certain delay. The scheduling process is done dynamically every TTI. The BS in return allocates dynamically some resources to the UE with a grant to which UE

10. In Huawei, srsLTE, and OpenAirInterface network configurations respectively

responses with a Buffer Status Report (BSR) to report the volume of data awaiting to be transmitted and some user data if it leaves some space in the TB after the control plane elements and signalling. SR and BSR mechanisms main properties are summarized in Table 1.1.

Technical specifications do not explicit the link between the BSR, the SR, QoS and the UG as they do not explain how the BS should take into account the BSR and the SR internally. However, we understand the BSR is used internally by the BS to maintain a buffer status estimation that should be used in scheduling operations.

## 1.2 Internet Protocol Suite

The Internet protocol suite commonly uses HyperText Transfer Protocol (HTTP) [68] as application layer, Transport Layer Security (TLS) [69] as security layer, Transmission Control Protocol (TCP) as transport layer and IP [70] for network layer communications. The delivery of data packets between client and server endpoints is ensured by the transport layer protocol. By 2022, TCP is largely dominant worldwide [21] with $\approx 90\%$ of the traffic followed by Quic, $\approx 8\%$ (30% in mobile networks [11]) and User Datagram Protocol (UDP) ($\approx 1.5\%$). TCP transports the biggest part of the internet HTTP traffics including web browsing, video streaming, communications and document sharing [8]. Quic [71] in the next decade should replace TCP with HTTP/3 [72] for the most of these usages. UDP is a datagram transport protocol used by other protocols (e.g. Quic) which is ultimately not so used for user traffic except for latency-sensitive real-time applications.

### 1.2.1 Transmission Control Protocol (TCP)

TCP [73] is a reliable transport protocol which uses Acknowledgment (ACK) and retransmissions to ensure the reception of all bytes. TCP is connection-oriented which mean that a full-duplex communication between endpoints is established after a handshake procedure (exchange of SYN, SYN-ACK, ACK packets) and closed at the end of the transfer (with FIN and FIN-ACK packets). To keep the communication reliable, TCP implements packet loss detection and recovery mechanisms. TCP also adopts to control the congestion (TCP must use a congestion control algorithm [74]) of the link with a continuous estimation of path capacity to avoid losses. That is the condition to avoid saturating network and losses due to a buffer overflow.

### 1.2.2 Congestion control

The Congestion Control Algorithm (CCA) is in charge of the congestion control and is able to do it using the feedback loop providing by the return ACK flow [75]. This feedback loop is illustrated in Figure 1.7. Acknowledgement packets initially commit the correct reception of a packet by the client. Thereafter, the transmission control continues the transfer by sending a new data packet. The volume of data "in-flight" in the network

11. https://www.rcrwireless.com/20220211/opinion/readerforum/heres-why-quic-is-still-worrying-mobile-operators-reader-forum

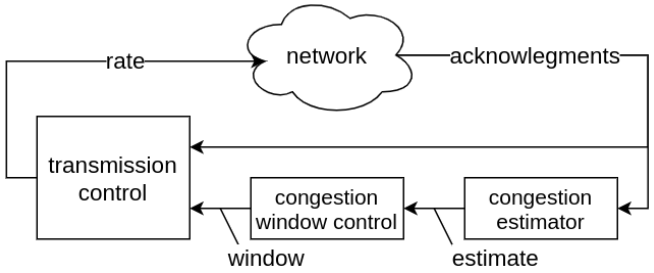



Figure 1.7 – Acknowledgments for TCP congestion control.

is limited by a *Congestion Window (CWND)* which must maintain a good usage of the network without saturating it.

In addition to report packet loss in the network, the ACK flow carries implicit hints on the network congestion [76]. Implicit hints include the measurement of the RTT, the variation of the RTT over time and the traffic transmission pattern. Congestion Control Algorithm (CCA) is composed of a congestion estimator and a congestion window controller. It must adapt the congestion window to the network conditions throughout the transfer duration. There are several versions of TCP which implement different congestion control algorithms [77]-Table II. Among them, CUBIC and Bottleneck Bandwidth and RTT (BBR) are very common on the internet ($> 60\%$ [78]). The Cubic algorithm (default CCA in Linux) estimates the available network path capacity from data packet losses. BBR initially developed by Google, uses a combination of RTT and bottleneck bandwidth estimations to estimate network path capacity. Research and developments around BBR are very important, and some experts announce that in the near future BBR will take over from cubic without replacing it [79]. Artificial Intelligence (AI)-based CCA (e.g. [80]) and CCA adapted to mobile networks [81, 82] are also active research topics.

### 1.2.3 Quic

*Quic* [71] is the next generation of transport protocol for internet services [83, 84]. Quic is based on decades of research and experiences of internet transport protocols. It meets the needs for more efficient internet transmissions, enhanced security and user privacy. Especially, the security aspect is put at a high level for Quic with a total packet encryption, including protocol header, that is a major change compared to TCP. It ques-

tions the utilization of middle-boxes in the network for traffic protocol optimizations [12] (Performance Enhancing Proxies [85]). Quic must be able to replace TCP on the internet to join evolution of user needs and latest network innovations. It already accounts for a significant portion of the Internet traffic [8]. Just as TCP, Quic guarantees the reliability of user data transport but with a revisited acknowledgement mechanism [86]. A CCA is also used following a similar model to TCP with BBR implemented by default for a number of protocol stacks [13].

12. TCP Optimization Enhancements for a Better Mobile Experience, F5 (2017) : `https://www.f5.com/company/blog/tcp-optimization-enhancements-for-a-better-mobile-experience`

13. List of open-source Quic stack implementations : `https://github.com/quicwg/base-drafts/wiki/Implementations`

## 1.3 Latency in Networks and Mitigation

In this section, we discuss latency in networks and their mitigation with a special focus on latency in mobile networks. Previous Sections 1.1 and 1.2 respectively introduced mobile networks technologies and Internet protocols. Latency is studied and defined in 3GPP documents for mobile networks and in Request For Comments (RFC) documents from transport area working group of Internet Engineering Task Force (IETF) for the Internet area [14]. The definitions of latency found in standards are presented in subsection 1.3.1. The subsection also discusses the distribution of end-to-end latencies. Afterwards, we present in subsection 1.3.2 the well-known network *bufferbloat* that is an important latency issue in networks, in particular in mobile networks, and which is an active research topic. In subsection 1.3.3, is presented the techniques of mitigation for latency implemented in 5G and proposed in the literature.

### 1.3.1 Definition of latency

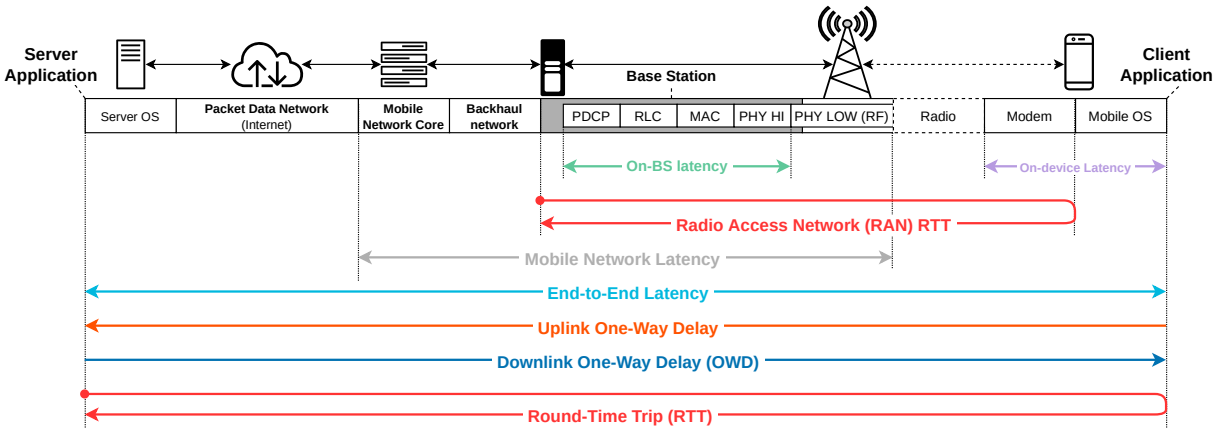



Figure 1.8 – End-to-End latency segments.

**Latency** 3GPP TS 22.261 [87] defines *end-to-end latency* as "*the time that takes to transfer a given piece of information from a source to a destination, measured at the communication interface, from the moment it is transmitted by the source to the moment it is successfully received at the destination*". It corresponds to the definition given in RFC7679 [88] of the One-Way delay metric where the given piece of information is an

14. In particuler, see IP Performance Measurement group (IPPM) IETF group [15]

IP packet. In Figure 1.8, end-to-end latency does not differentiate the channel, uplink or downlink. The latency associated to a path direction is called *One-Way Delay (OWD)* on the figure and can be of uplink for latency from mobile to IP network or downlink for latency from IP network to mobile.

5G RAT Key Performance Indicators (KPIs) are defined in 3GPP TS 38.913 [89]-7. Especially it defines the *control plane* latency, the *user plane* latency and the *latency for infrequent packets*.

The control plane latency is understood as the time to move from idle to active state (ready to transfer data) for a UE (i.e. the time to effectuate Random Access procedure Channel (RACH) and initial access procedure Appendix A.7.6).

The RAN One-Way Delay (OWD) is defined as "*the one-way transit time between a packet being available at the IP layer of the UE/RAN edge node and the same packet being available at the RAN edge/UE. The RAN edge node is the node providing the radio access network interface towards the core network*". The RAN RTT is the sum of uplink and downlink RAN OWDs. Generally, when a latency for a 5G system is given, it refers to the RAN RTT.

Latency for infrequent small packets is a mix of RAN OWD and control plane latency because it is the time it takes to successfully deliver an application layer packet from the mobile device to the egress point in the RAN when the device starts from its idle state.

The *mobile network latency* is the latency of the network nodes serving UEs, i.e. base station, backhaul network and mobile network core, excluding the air interface and mobile. In Fig. 1.8, on-BS latency and on-device latency are also shown as latency segments.

The *Round-Time Trip (RTT)* is the delay for a packet to reach from the source to the destination and a response back to the source. RTT is usually measured with ping [90] and is equal (subject to some uncertainties due to the method of measurement with a program) to the sum of uplink and downlink OWDs. Network RTT is a KPI of the network since it measure the latency experienced by a user while it interacts with a distant machine via the network.

**Network segments** Each network segment on the path illustrated in Fig. 1.8 comes with different types of delay regarding its property. There are queueing delay due to buffering, propagation delay due to physical limits of the information speed ($2.10^5$ km/s in a fiber, $3.10^5$ km/s in the air), processing delay due to network processing and error recovery delay. For instance, server OS segment consists in on-chip processing delays,

latency of RLC layer segment inside BS is dominated by queueing delays and latency of radio interface, bounded by the TTI duration. In total, the *mobile network latency* is composed of:

$$T = T_{\text{RAN}} + T_{\text{Backhaul}} + T_{\text{Core}} \tag{1.2}$$

Where $T_{\text{Backhaul}}$ is the latency of the backhaul network (mostly transport latency), $T_{\text{Core}}$, the latency of the core network (associated to routing latency and transport to PDN) and $T_{\text{RAN}}$, the latency introduced by the RAN, and the focus of this thesis.

Traditionally, the effects of quick varying radio channel capacity and transmissions losses in the RAN are handled using multiple packet buffers with different latency characteristics.

— A transmission buffer which stores arriving data packets to be transmitted. It is made large in mobile networks to take advantage of new available radio capacities in a millisecond timescale and conversely to avoid buffer overflow packet flow when the available capacity shrinks.
— An handover buffer to proceed to device mobility from one base station to another seamlessly without losses.
— An aggregation buffer to collect data from different bearer and generate a TB in each transmission opportunity.
— Retransmission buffers related to MAC HARQ and RLC ARQ processes.
— An in-sequence buffer to avoid excessive reordering.

**Jitter** Until now, we have defined the absolute latency of an individual packet.

The *Packet Delay Variation (PDV)* is given for a pair of packets within a packet flow between 2 measurement points. In other words, the PDV is the difference between the OWD of a selected pair of packets (RFC 3393 [91]). PDV is also called *latency variation* or *jitter*, even if the last designation is viewed as confusing [91]-1.1. ITU-T document Y.1540 [16] (2019) §6.2.4 calls it *2-point packet delay variation* and gives the preferred method for summarizing the delay variation based on quantile. Quantile-based limits of PDV method is to select upper and lower quantiles of the delay variation distribution and then measure the distance between those quantiles.

In the following, we use this method with the $10^{\text{th}}$ percentile and the $90^{\text{th}}$ percentile as illustrated with the Cumulative Distribution Function (CDF) of latency for a given connection in Fig. 1.9. It corresponds to the majority of the measurements. In the figure,

16. https://www.itu.int/rec/T-REC-Y.1540-201912-I/en

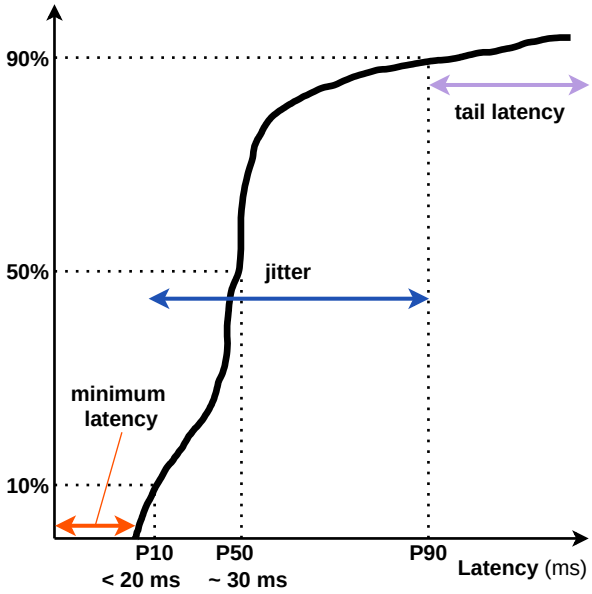



Figure 1.9 – CDF of latency for a given connection and definitions

we indicate *minimum* latency, which is the minimum OWD observed for a given segment and *tail latency*, also known as *high-percentile* latency, which is high latencies that packets experienced infrequently.

Jitter is an important metric for the buffer sizing [92] and to determine the dynamics of queues withing a network. Each segment of the network generates more or less jitter. For instance, in-sequence delivery of PDCP causes "flushing" effect that generate lot of latency variation while a non-congested processing function generates latency but not latency variation.

**Summary** Three metrics are used to characterize a network segment, the OWDs, the RTT and the jitter. Users experienced the end-to-end latency, composed of a series of latency with different characteristics introduced by each segment on the network path. This thesis focus on the latency of the RAN.

### 1.3.2 Bufferbloat

Some network segments latency are dominated by the queueing delay (or buffering delay). Queueing delay appears when packets arrive at a network node at a higher rate than the node can process them and puts them into a buffer. It is a key component of delay in packet switched network.

It has been observed by Gettys et al. in [19] (2011) that today's computer networks are suffering from unnecessary latency and poor system performances. They define the *bufferbloat* (i.e. excess queueing latency) phenomenon, which is the existence of excessively large and persistently full buffers inside the network. Bufferbloat is a major concern of the research since then to unleash network performances. Optimal buffer sizes should be equal the Bandwidth-Delay Product (BDP), however, as the delay is hard to estimate, larger buffers have been deployed (the problem of buffer sizing) to raise throughputs and prevent losses. It has favored the appearance of bufferbloat.

It is prevalent on last-mile access network (e.g. LTE, Wifi.) [93] since this network segment aggregates multiple flows and have recourse to large buffer because of the channel capacity variation and the needs of error recovery. An important work on bufferbloat in Wifi has been done by Toke Hoeiland-Joergensen in [93]. Especially, it indicates bufferbloat is very real and common in radio access network and tends to be significant, most often on the order of several hundreds milliseconds (more than 20000 km equivalent in an optical fiber).

A direct solution to tackle bufferbloat is to control dynamically to maximum queue size instead of letting it fills up. Hoeiland as many research works proposes aqm [94, 95] solutions[96–99] where queues are not yet considered as infinite First-In First-Out (FIFO) queues but implement a packet dropping policy (e.g. Adaptive Random Early Detection (ARED), Controlled Delay (CoDel) [100], Proportional Integral controller Enhanced (PIE) [101]).

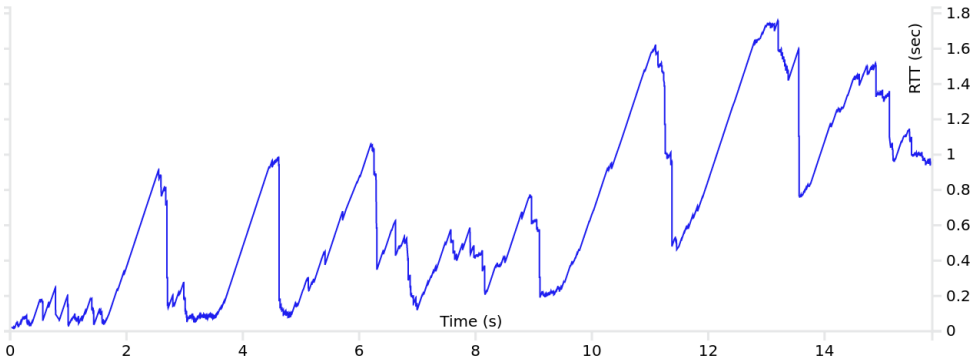



Figure 1.10 – Packet delay exhibiting bufferbloat.

Bufferbloat is also prevalent in cellular networks [102][103]-2 and causes many problems for low latency services but also for Transmission Control Protocol (TCP) performances [103]-3. The bufferbloat is concentrated at the bottleneck, therefore at the level

of the Radio Access Network (RAN). In the downlink path of RAN, it is the Radio Link Control (RLC) transmission buffer which stores the packets before they are multiplexed and transmitted over the radio channel.

Figure 1.10 depicts the Round-Time Trip (RTT) over time of a TCP transfer in a commercial mobile network. At the beginning of the transfer, the RTT is close to the end-to-end $RTT_{min}$ ($\approx$ 30 ms). Thereafter, the RTT increases with some peaks due to the packet scheduling in the RAN but never reduce with a steady latency of 1 second, which is considerable. The data volume in-flight in the network reaches a maximum of 1.6 Mbytes (= 1100 Internet Protocol (IP) packets) without packet losses. Thus, we estimate the RLC buffer associated to one Data Radio Bearer (DRB) (i.e. one User Equipment (UE)) is at least 1.6 MB, it is ten time larger than the Bandwidth-Delay Product (BDP) that is equal to $5.375 \times 0.03 = 0.161$ MB [17]. However, a large buffer is necessary to handle rapid fluctuation of the radio channel capacity (no starving when new network capacities are available), and transmission loss (when the network capacity is degrading).

Bufferbloat tackling is a complex challenge [104] and an important issue for cellular networks performances [6] not only for low latency services but also for transport protocols.

17. Cell capacity is 43 Mbps and minimum RTT 0.03 sec

| **Delays** (from [105] chapters) | **Delay type** P | V | Q | S | E | I | H | **Techniques** | **5G mechanisms** and related research | **Related studies** for instance |
|---|---|---|---|---|---|---|---|---|---|---|
| **Structural** (§II) | | V | | | | | | Content placement | Mobile Edge Computing | [106–109] |
| | | | | | | | | Virtualization | Cloud-RAN | [27, 110–112], [28]-13.7,[113]-VI |
| **Signal propagation** (§IV-A) | | V | | | | | | Optimized propagation line | AAS, CoMP | [114, 115] |
| | P | V | | | | | | Short transmission time | Frame structure | [116–120],[121]-III.A, [122]-IV.A,[24]-IV.A |
| **Medium acquisition** (§IV-B) | | | | | | I | | Connection establishment | Optimized Random Access procedure | [123, 124],[125]-2,[126]-IV.A |
| | | | | | | I | | Session re-establishment | RRC IDLE mode | [127] |
| | | | | S | | | | Air interface access | Scheduling timing | [128] |
| **Serialization** (§IV-C) | P | | | | | | | Concatenation and Segmentation | Refactoring of in-sequence delivery and PDU segmentation | [129]-7.1.2,[130] |
| **Link error recovery** (§IV-D) | | | | | E | | | Error detection and Recovery | Optimized IR-HARQ and | [131–133],[134]-VIII, |
| | | | | | E | | | | shorter HARQ ACK feedback | [129]-7.1.1 |
| **Switching and Forwarding** (§IV-E) | | | | | | | H | Inter-cell communication | Handover procedure | [135],[129]-7.2.2, [126]-IV.C |
| **Queuing** (§IV-F) | | | | S | | | | Flow and circuit scheduling | Slicing and QoS flow | [27] |
| | | | Q | | | | | Smaller network buffer | RLC buffer sizing | [136] |
| | | | | S | E | | | Packet Scheduler | Agile scheduler and flexibilites | [50],[24]-V.A, [122]-IV.B,[126]-V.B |
| | | | | S | | | | | Grant-free access | [137] |
| | | | Q | | | | | Traffic shaping and policing | Policy Control Function (PCF) | [138] |
| | | | Q | | | | | | L4S for 5G | [139–141] |
| | | | Q | | | | | Queue Management | Active Queue Management (AQM) | [22, 92, 136, 142–149] [150, 151][152]-VIII.B |
| **Insufficient capacity** (§V-A) | P | | Q | | | | | Leveraging interfaces | LDPC coding, higher MCS, higher frequency, wider bandwidth, Carrier-Aggregation | [153–155] |
| | | | | S | E | | | Multiple links | Massive-MIMO, multi-connectivity | [114, 156–158] |
| **Under-utilized capacity** (§V-C) | | | Q | | | | | Congestion control sensing | ACK feedback loop | [76, 85, 159–161] |
| | | | | | | | | | Assisted capacity sensing, ECN | [162–164], [161, 165] |
| | | | | | | | | More aggressive congestion control | Slow-start, CCA | [81, 82, 166–168] |

Table 1.2 – Types of delays and techniques for reducing latency in 5G *(adapted from [105] for mobile network case). P: processing, V: propagation, Q: queueing, S: scheduling, E: link error, I: medium acquisition, H: handover.*

### 1.3.3 Latency Mitigation in 5G

*It's time for low latency in cellular networks.* Briscoe et al. in [105] (2016) investigate techniques for reducing internet latency. In [105]-Fig. 2 they propose a classification of source of delay and techniques for reducing latency. They identified 5 types of delays which are: §II) *Structural delays*, related to network architecture; §III) *Interaction between endpoints* (transport and session initialization, E2E packet loss and recovery); §IV) *Delays along transmission paths* which cover propagation delays, link error recovery, queueing delay, medium acquisition... ; §V) *Delays related to link capacities* such as insufficient capacity or *under-utilized capacity*; §VI) *Intra-end host delays*, mostly operating system and network stack delays.

We use their classification to establish Table 1.2 which summarizes 5G mechanisms and related research for latency reduction in mobile networks. Some related studies are given in the last column. We do not present a comprehensive survey of low latency for 5G. To going further, [24] (2021) gives an overview of low latency for wireless communications with an evolutionary perspective from 2G to 5G. [15] (2018) is a comprehensive survey on low latency for 5G with an end-to-end analysis, RAN, core, network and caching. [169] describes Time-Sensitive Networks (TSN) and Deterministic Networks (DetNets) standards and related 5G research for 5G ultra-low latency networks.

5G system is designed to go towards low latency transmissions, 10 ms for best effort bearer and 1 ms for time-critical bearer. To ensure that, 5G employs both end-to-end and RAN tools.

**End-to-end tools in relation with RAN** End-to-end toolbox includes the traffic differentiation, the network softwarization, the content placement and the network architecture and management [170]. The new 5G Quality of Service (QoS) framework introduces the end-to-end QoS flow architecture where packets are carried in virtual circuit associated to a QoS class (e.g. Default best-effort, latency-optimized, service-optimized, etc.) Differentiation of traffics allows the Medium Access Control (MAC) scheduling to prioritize a flow over another and radio resource reservation (see Table 1.2-"Flow and circuit scheduling"). It comes with the network *slicing* which enables a virtual isolation of end-to-end network on the same physical network infrastructure.

Slicing [27] comes from the concepts of software-defined networking (SDN) [171] and network function virtualization (NFV). In practice, Slicing is the reservation of a bandwidth part to a slice for a specific service. QoS flows and slicing are useful tools for

latency-sensitive latency scheduling.

Cloud-RAN (C-RAN) [110, 112] allowed by network softwarization redefines the Base Station (BS) network functions placement in a pool to share the same infrastructure (see Table 1.2-"Virtualization"). For instance, handover between base stations is facilitated with a shorter handover latency. Moreover, BSs higher layers and control plane could be placed in the same datacenter facility as the 5GC to reduce control place latency.

There has been a trend in recent years to bring content closer to clients with copy of the content stored in a decentralized way e.g. with Content Delivery Network (CDN) (see Table 1.2-"Content placement"). This network architecture is completely taken into account in the design of the 5G architecture. The content could be now co-located in a data centre with the core and the cloud-RAN. Mobile Edge Computing (MEC) [18] redefines the content and computing location closer to the client, reducing efficiently latency [106]. In this architecture, the packet path is shortened to the RAN and a backhaul network. However, even if this architecture could be employed for all type of traffics without traffic differentiation (they are using IP routing), the deployment cost is very high, limiting its utilization for only few latency-specific services (e.g. cloud gaming, XR [107]).

**RAN tools** Radio Access Network (RAN) segment counts for the larger part of latency in the mobile network with a baseline RTT of 20 ms in LTE for a total mobile network latency usually around 30 ms. Third Generation Partnership Project (3GPP) published several documents in 2016 where latency reduction is studied for LTE [172]. Some practical recommendations have been given concerning network configurations for low latency [26]. These studies inspired the development of New Radio (NR). 5G-NR Radio Access Technology (RAT) is able to tend towards low latency thanks to key features [1] that distinguish it from LTE. Latency we found in RAN are of different types: 1) Processing (i.e. service time in queueing theory) delay; 2) Propagation delay; 3) Queueing (or buffering) delay; 4) Scheduling delay (i.e. insufficient capacity); 5) Link error recovery delay; 6) Medium acquisition delay; 7) Handover latency. In table 1.2, paragraphs below are put in relation with delays and techniques identified by [105] in the column "Paragraphs".

**Processing delay** Delays associated to data processing functions at every layer of the radio protocol stack (Fig. 1.2).

18. Redhat documentation : `https://www.redhat.com/en/topics/edge-computing/what-is-multi-access-edge-computing`

The data serialization (Table 1.2-"Concatenation and segmentation") has been revisited for Packet Data Convergence Protocol (PDCP) and RLC layers to avoid latency due to segmentation and in-sequence delivery. In-sequence delivery means a fully decoded Protocol Data Unit (PDU) with a sequence number $N$ could not be delivered until the $N-1$ PDU are not fully decoded and transmitted. Thus, a retransmission at a lower layer of an old PDU might block the transmission of several PDUs. In-sequence has been removed from PDCP layer (since after PDCP, the segmentation is handled by IP layer for user-plane) and revisited for RLC layer. Especially, RLC entities are not allowed anymore to perform concatenation of multiple PDCP Service Data Units (SDUs) into one PDU since the MAC is already in charge of multiplexing multiple RLC PDU. This simplifies the PDU decoding at the counterpart with a low impact on achieved Transport Block (TB) utilization rate and control overhead.

Low-Density Parity-Check Code (LDPC) (user-plane) and polar codes (control-plane) for data encoding replaces the LTE turbo-coders with a gain in terms of spectral and computational efficiency (Table 1.2-"Leveraging interfaces"). LDPC in particular changes the transmission scheme by introducing more flexibilities. As in LTE, in NR as well a huge TB is split into multiple Code-Blocks (CBs), but NR introduce the concept of Code-Block Group (CBG) that are coded and decoded independently without prior buffering. As soon as a CB is decoded, it is transmitted to MAC layer to reassemble the transport block. Since the TB structure has been modified to put headers inside the TB instead of all at the beginning, a block of data could be transmitted as soon as it is decoded when in LTE we had to wait the decoding of the whole TB (which could be huge in 5G) before transmitting all blocks of data. More efficient coder/decoder and early transmission of CBs, the Transmission Time Interval (TTI) could be shortened below 1 ms.

In general, the current hardware performs better than when LTE was launched, which enables a reduction of the processing times defined in the standard. The processing time for data encoding and decoding defined as 3 ms in the two cases in LTE is now more flexible and lowered to one TTI duration in 5G.

**Propagation delay** The time of propagation is bounded to the TTI duration, it is the time to fully receive a transport block. The TTI of 1 ms endorses the propagation of the radio wave in the air (300 km per ms), the physical transmission system delay (i.e. antenna) and the decoding time at the Physical Layer (PHY) layer. The adding of multiple numerologies in 5G enables transmission in a fraction of millisecond (§1.1.2.2,

Table 1.2-"Short transmission time"). There are also a propagation delay in the backhaul network (using fiber) to carry data packet to the network core and after that in the Packet Data Network (PDN). A closer content palliates to an important propagation delay in the mobile network.

**Queueing delay** is the most important source of latency and occurs when end hosts transmit more packets than the RAN can sustain. The bufferbloat presented in §1.3.2 is the most illustrative bad consequence. There are several ways to mitigate the congestion-related delays in the RAN. The queueing management provides an efficient solution to queuing delays (Table 1.2-"Smaller network buffer","Queuing management"). RLC transmission buffer is usually designed large to handle rapid fluctuation of radio channel with the drawback to generate an important queueing latency when it is full. A research direction is to adapt the RLC buffer size dynamically to the channel conditions. Another promising research direction is the utilization of AQM systems combining queueing management, end hosts and congestion signalling (e.g. Explicit Congestion Notification (ECN) [165]) for cellular networks.

The most active project on this subject is Low Latency, Low Loss, Scalable throughput (L4S) applied to cellular networks. L4S [139] is a method that provides fast indication of congestion from RAN, scalable end hosts (Congestion Control Algorithm (CCA) which reacts proportionally to congestion hints) and a queueing management based on 2 queues (one for L4S capable flows and one non-L4S capable). L4S comes from the RITE project [19], a project funded by the European Union for reducing internet latency. L4S is well discussed in the literature and in Internet Engineering Task Force (IETF) working groups and is the subject of studies to be applicable to 5G networks like in Bruhn [173] (2020).

In the same idea, the cooperation between end points and network is a promising direction to mitigate congestion and fully utilized radio channel capacity. For that, adapted congestion control and explicit congestion hints are employed (Table 1.2-"Under-utilized capacity").

**Scheduling delay** is related to the multiple aspects of cellular networks. Principles behind the scheduling improvements are the reduction of the access latency, the fastest possible adaptation to fluctuations in the radio channel and the wider bandwidth for increased radio channel capacity (Table 1.2-"Packet scheduler","leveraging interfaces").

19. RITE project webpage : `https://riteproject.eu/dctth/`

We presented in §1.1.3.2 the different radio resource allocation schemes with grant-free and grant-based access. The L2 pre-emption reduces drastically the access latency by quickly schedule an urgent latency critical packet with a short TTI that overwrites another ongoing downlink transmission. DL Semi-Persistent Scheduling (SPS) and Fast Uplink Grant (Fast-UL) reduces also latency by cancelling the needs of waiting for a data arrival (DL) or the reception of a Scheduling Request (SR) (UL). Fast-UL is considered as an important factor of access latency reduction in the uplink, but it comes with the problem of the estimation of transmission needs.

Grant-based dynamic scheduler benefits from radio frame structure flexibilities (e.g. OFDM-symbol level scheduling), higher numerologies (e.g. shorter TTI), duplex scheme (e.g. full-duplex transmission scheme) and flexible scheduling timing (e.g. timing between grant and transmission) [50] to reduce access latency for a wide range of users. All of these combined with a more efficient channel sounding and signalling, schedulers will be more agile to adapt quickly to channel variations. Finally, with Advanced Antenna System (AAS), massive Multiple-Input Multiple-Output (MIMO) [10]-II.C, beamforming, wider bandwidth, higher Modulation and Coding Scheme (MCS) and higher frequencies, schedulers will be able to leverage interface capacity and mitigate insufficient capacity in all circumstances. Several propositions of scheduler already show how to use efficiently newly available radio capacities, especially for mmWave radio channel [10]-II.B.

**Link error recovery delay** Link error recovery are mostly handled by the Hybrid-ARQ (HARQ) process. 5G uses HARQ retransmission procedure using incremental redundancy to improve successful transmission decoding probability with the soft combination of information from multiple transmission attempts. Also, retransmissions are supported on a finer granularity with CBG, limiting the impact of an error on a part of the transport block at few CBG instead of the entire TB. The HARQ ACK/NACK timing is flexible of 1 to 15 instead of fixed 4 TTI in LTE. The asynchronicity of HARQ is an important difference in error recovery compared to LTE and an important enabler for reducing radio link latency. It was common in LTE to see bursts of data after 8 ms of pause because of an HARQ retransmission. Retransmissions in 5G should be less visible on latency Empirical Cumulative Distribution Function (ECDF). Research is still active on HARQ with multiple proposed method to increases the recoverability of data [132].

The RLC Acknowledged Mode (AM) entity takes a part in the remaining radio link recovery after HARQ process with the RLC Automatic Repeat reQuest (ARQ) process.

In this task, [130] shows the impact of acknowledgments on application performance in LTE and propose to adjust RLC timers in §5.3 to limit this latency. [174] discusses the desirability of removing ARQ at the RLC layer for latency-sensitive traffic by using an Unacknowledged Mode (UM) entity instead of AM entity. It could be made only if traffic flows are differentiated between those loss-sensitive and those latency-sensitive.

**Initial access delay** The initial access is the procedures by which a UE established the connection with a cell through the Random Access procedure Channel (RACH) procedure and the a Radio Resource Control (RRC) session is established/re-established, ready to transmit and receive data. The latency reduction for the initial access delay is mainly related to the RRC session re-establishment and an optimized state machine transition from inactive to active with the arrival of the intermediate IDLE state.

**Handover latency** Mobile device often moves from one cell to another when it is moving. Handover has been identified as an important source of latency spikes during the transfer of a base station to the other [129]-6.2. Handover procedure begins with control signalling exchange following the re-routing of the flows from the mobile core and the transfer of the PDCP transmission buffer from the old BS to the new one.

The increased cooperation between BSs and multiple attachments makes the handover procedure smoother for the mobile users, limiting latency spikes. Also, the physical co-location of BS higher-layer network functions in the same pool will decrease the transfer latency of one BS to another.

**Summary** 5G-NR RAT introduces many improvements for reducing latency in mobile networks at RAN and end-to-end levels. Latency reduction in RAN will come from an optimized processing time, a better link error recovery handling, a dynamic and flexible scheduler, and a work to tackle queueing latency. The objectives are both to reduce the baseline RAN RTT and to contain high latencies inherent to cellular radio networks (i.e. bufferbloat). 5G gives tools to operate low latency networks, but, a lot of research work and experiments remain to be done to effectively brings achieve low latency. We give the example of the complex agile dynamic scheduler or the Active Queue Management (AQM).

Chapter 2

# Research Method

**Abstract** : *In the computer networking field, the research methods used are simulation, emulation and experimental trials in commercial networks. This chapter presents the research method employed for this thesis based on a lab testbed. We set up a lab testbed utilizing the OpenAirInterface (OAI) project that reproduces a Radio Access Network (RAN) with a LTE Base Station and 2 commercial smartphones. The network architecture is designed to reduce non-RAN related latencies. Traffic generation methods are then presented with the transport protocols and endpoints applications. Finally, the latency analysis tools and methodologies are explained and reveal a difficulty to catch in-RAN latencies at layer and packet-level.*

## Contents

**2.1 Objectives . . . . . 44**
**2.2 Lab testbed with OpenAirInterface . . . . . 46**
2.2.1 Comparison of RAN solutions . . . . . 46
2.2.2 Mobile network core . . . . . 48
2.2.3 Radio Access Network testbed . . . . . 49
**2.3 Traffic generation . . . . . 52**
**2.4 Analysis tools and methodologies . . . . . 55**
2.4.1 Active RTT measurement . . . . . 55
2.4.2 Passive measurements . . . . . 56
2.4.3 Network captures synchronization . . . . . 58
**2.5 Conclusion . . . . . 60**

## 2.1 Objectives

This thesis is written in the field of *computer networking*. This field can be viewed as being comprised of three different aspects namely mathematics, engineering and science. Mathematical aspect provides theoretical foundations in particular of what is achievable and computable. Computer networking field has always been throughout its history linked to engineering. A huge engineering work has been necessary to build physical part of computer networks in a sense that internet is probably the most important and complex machine build by the humankind. Software engineering is now everywhere in the network developments especially when we consider the tendency of network softwarization. Scientific method is important in computer when the network itself can become the subject of study.

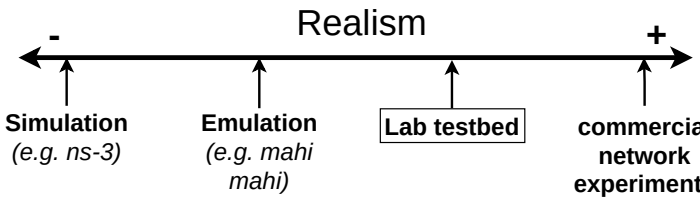



Figure 2.1 – Degree of realism of the different research methods.

In networking, the scientific studies are usually based on one of the three following approaches (see Fig. 2.1): 1) Simulation where the network is entirely simulated with a computer program; 2) Lab experiments where the system under study is placed in a controlled environment with sometime emulation for non-essential parts; 3) Real-world experiments where the system under study is placed in a representative environment. The research presented in this thesis takes a use of three methods [175] with always the idea that observations and ideas developed are as close as possible to real world systems.

The research process we followed in this work is primarily based on a lab experiment approach, complemented by simulation and commercial network experiments when necessary. The lab testbed we present in the next section is a trade-off between the realism of a commercial network experimental setup that represents customer experience of the networks and a computer simulation where the mobile network is modeled. In fact, too often work based only on simulations are finally never deployed on the internet because idealisations often leading to very different behaviour on networking devices.

The objectives of the lab testbed are multiple:

1. It should exhibit the issues related to latency in radio access networks, completed by the literature and the commercial network trials;
2. It should permit a deep and comprehensive analysis of the latency root causes, especially, it implements probes and capture points.
3. It is used as a support for the evaluation of the proposed solutions.

The testbed reproduces a cellular radio access network (*i.e. mobile terminals, base station*) completed with a mobile network core and a local server to close the path between the client and the server.

The following Section 2.2 presents the lab testbed setup aspects. Section 2.3 describes the application which generates the network traffic. The last Section 2.4 of the chapter explains the analysis tools and methods for latency analysis in use in the testbed.

## 2.2 Lab testbed with OpenAirInterface

In this section, we describe the main testbed used in this thesis. It makes a link between real-world cellular networks and simulations. In one hand, the proposed experimental setup is a white box for latency analysis in contrast to deployed networks. On the other hand, ideas developed on a real radio interface ensures that they can be implemented in real systems (representativity of experiment in contrast to simulations). For these reasons, an open-source software cellular network is the most suitable solution. We chose to use OpenAirInterface (OAI)[176][2] as the base of the presented lab testbed.

A quick review of the existing solutions for cellular network research is proposed on the next section followed by a presentation of testbed and finally a brief technical introduction to OAI radio air interface.

### 2.2.1 Comparison of RAN solutions

| Solution | LTE | 5G | Open* | Advantages | Limitations |
|---|---|---|---|---|---|
| **Amarisoft** | ✓ | ✓ | ✗ | •Already deployed | • Closed sources<br>• Expensive |
| **Radisys** | ✓ | ✓ | ✗ | • Advanced open-RAN | • Closed sources<br>• Expensive |
| **srsLTE** | ✓ | ✓ | ✓ | • Stable • Modularity<br>• Good code quality | • Small community |
| **OpenAirInterface** | ✓ | ∼ | ✓ | • Active project,<br>• Research oriented,<br>• Large community | • 5G developments On-going<br>• Code quality |

Table 2.1 – Solutions for cellular network research testbed. **Open: Open-Source.*

The lab testbed is a research platform designed around a private cellular network. We considered 4 solutions of private cellular network presented in Table 2.1. An evaluation of different solutions, especially of OAI, is provided in a recent work of Vilakazi et al. [177]. **Amarisoft** [1] provides LTE BS (BS), 5G BS (gNodeB (gNB)) and UE software which run on off-the-shelve hardware. The callbox is an easy-to-deploy closed linux machine on

1. `www.amarisoft.com`

which runs a gNB and a core network. It is designed for LTE and 5G testing of user equipment with advanced configurations. Amarisoft solution is a very powerful machine for cellular network research.

In the same idea, **Radisys** [2] offers a complete 3GPP compliant RAN software. Radisys 5G software runs also on generic hardware and claims to have open interoperable interface.

**srsLTE** [3] [178] is one of the most important open software for mobile wireless network. It implements a complete LTE chain in an open-source way [4]. A very good point to srsLTE is the code maturity. However, at January 2020, there were no communication about a 5G release.

**OpenAirInterface** (OAI) [5] [6] [2] is an open-source software that proposes a complete implementation of LTE network and currently actively develop 5G Standalone (SA). Developments in the project are for a big part the contribution of the French Eurecom research laboratory, but it is also supported by important companies (*e.g. Fujitsu, Qualcomm, Nokia, Meta, Orange*), universities and organizations (*e.g. bcom, chinese academy of sciences, university of Kent*). OpenAirInterface Software Alliance (OSA) which drives the OAI project to make partnership with bigger open network project (*e.g. open RAN, magma core*).

De Javel et al. have presented in 2021 (after our initial evaluation) a new open-source 5G RAN, free5GRAN [179] which put priorities on modularity, and experiment reproduction for research and education purpose.

A common property of the presented solutions is the use of a generic hardware when the software is the major part of the work. Softwarization of networks is a long-term tendency by which network functions are moved from specific hardware and low software engineering to generic hardware and high software engineering. More generally, the layer 2 which ensures physical transmission in the Open Systems Interconnection (OSI) model is a software which runs on a generic computer. The physical layer 1 where bytes are converted into Orthogonal Frequency-Division Multiple (OFDM) symbols is rather implemented in a specific software-defined radio hardware.

The differentiation of the projects could be made on the advancement of the 5G RAT implementation and the open-sourceness of their solution as an enabler for research. The

2. https://www.radisys.com/solutions/openran
3. https://www.srslte.com/
4. https://github.com/srsran/srsran
5. https://openairinterface.org/
6. https://gitlab.eurecom.fr/oai/openairinterface5g

usage of an open-source project means that we could manipulate, read and add code to verify hypothesis, that is more interesting for research purposes than closed-source Application Programming Interface (API)-based project. Bringing all these arguments together, OAI corresponds the most to our use-case. In addition, the developer community around the OpenAirInterface project is very active with multiple side-projects, workshop and additional features for OAI. Finally, there is an experience at Orange of OAI as the company is an important member of the OSA and a long-time contributor to the project.

An important limitation of OAI is the late release of 5G in July 2021 and then, the 5G stack code maturity. In the following experiments and implementations, the LTE version will be used instead of 5G for these reasons. That is not a problem as we explained in paragraph 1.1.2.1, LTE and 5G have in common the same Layer-2 protocols and mechanisms. In many cases, a short discussion is sufficient to extrapolate LTE results to 5G. When a 5G feature is necessary, we could emulate it in the LTE BS's stack.

### 2.2.2 Mobile network core

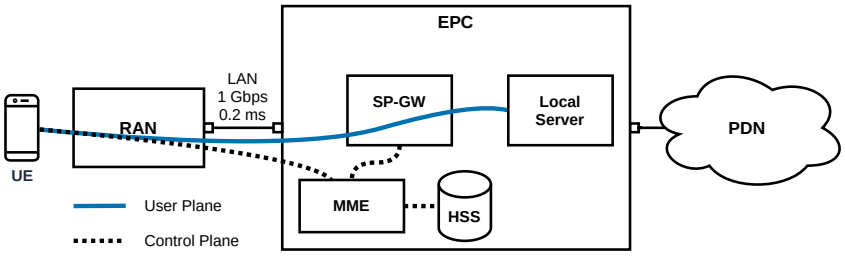



Figure 2.2 – Mobile network core architecture.

We are using the OAI Evolved NodeB (eNB) and Evolved Packet Core (EPC). EPC is consisting on 3 entities, Home Subscriber Server (HSS) for SIM card registry, Mobile Management Entity (MME) for paging and Serving Gateway (SGW) to connect to IP network. We do not pursue a deep understanding of EPC since our subject of study is the RAN mechanisms. In the following of this work, the focus on engineering aspects is put on the OAI base station as the main object of the radio access technology study.

We set up only one Best Effort (BE) bearer at the UE RRC connection establishment and do not consider any QoS bearers as it is a common configuration in operational networks. The EPC is made as transparent as possible, especially to perform RRC procedures.

### 2.2.3 Radio Access Network testbed

| Parameter | Value |
|---|---|
| Base Station | OAI eNodeB (lte-softmodem) |
| #UE | 2 |
| Antenna format | SISO |
| Air Interface quality | SNR > 12 dB |
| E-UTRA Band | 7 |
| Frequency | 2.56 GHz (UL) / 2.68 GHz (DL) |
| Radio frame structure | FDD |
| Bandwidth | 10 MHz |
| #PRB | 50 |
| Max. modulation | `16-QAM` (UL) / `64-QAM` (DL) |
| Max. achievable datarate | 17 Mbps (UL) / 37.5 Mbps (DL) |
| Radio slot duration | 0.5 ms |
| Radio subframe duration | 1 ms |
| HARQ max retx | 4 |
| HARQ retx delay | 8 ms |
| Radio Bearer QoS | Best-Effort |
| RLC mode | Acknowledged Mode |
| RLC `t-Reordering` | 32 ms |
| DRX | Not configured |
| Scheduling algorithm | Round-Robin / Proportional-Fair |
| SR periodicity | 10 ms |
| BSR periodicity | 64 ms |

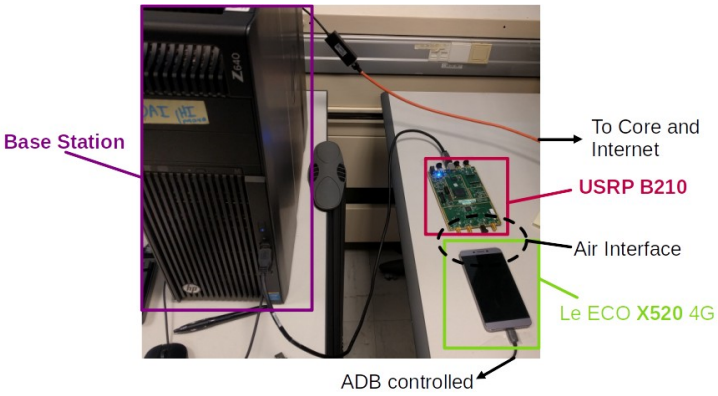
Table 2.2 – OAI lab testbed configurations.

#### 2.2.3.1 Experimental testbed elements

Alongside BS the RAN is consisting on 2 Commercial Off-The-Shelf (COTS) smartphones and a Software-Defined Radio (SDR) card Universal Software Radio Peripheral (USRP) B210 (Ettus). The photo in Figures 2.3 illustrates those elements. RAN configurations are summarized in Table 2.2.

#### 2.2.3.2 Base Station hardware

The USRP B210 is the key to bring real-time LTE communication to RAN with a capacity of 61.44 MS/s (*mega symbol per second*) far more than the capacity of the LTE




(a) Testbed elements.

(b) With the Radio-Frequency (RF) shield for a radio environment isolation.

Figure 2.3 – Photos of the OAI RAN testbed

radio interface capacity. It is based on a Field Programmable Gate Arrays (FPGA) chip connected to the host PC with USB 3.0. The x86 machine on which eNB runs has an Intel Xeon E5-2640 CPU with 16 cores clocked at 3.00 GHz and 16 GB of memory. The base station is connected to the core through a gigabyte Ethernet connection. The core is located in the same room as RAN on another machine to avoid network delays. Server applications are installed on the same machine as the core entities.

#### 2.2.3.3 User Equipment

We opted for a COTS smartphone as UE instead of OAI's soft UE. The first reason of that choice is the will to focus on the base station in this work. As we already argued, we think that an important research work has to be made at the base station and thus, we consider UEs as black boxes which are 3GPP standard-compliant. Also, a smartphone is a complex system on which different modules interact in such a way that they are not described in any standards. For instance, the LTE modem is of course compliant to 3GPP standard to communicate with LTE network. However, standard does not describe in detail some functions and then, they are implemented differently by founders. Also, modem has complex interactions with an operating system (e.g. Android or iOS) and drivers. The search for realism with the RAN testbed implies to use these commercial black boxes called smartphone instead of a less representative UE i.e. a Linux OS, x86 architecture, OAI driver and USRP modem.

The 2 models of smartphone we are using are:

— *LeEco Le 2*[7] (2016) smartphone running rooted Lineage 16 OS (derivative from Android 9) and communicating in LTE with Qualcomm snapdragon 652 modem.
— *Xiaomi Mi 10T 5G*[8] (2021) smartphone running rooted Android 10 and communicating in LTE and 5G with Qualcomm 865.

The smartphone embeds all the necessary to generate and receive data and produce logs. Smartphones are controlled with the Android Data Bridge (ADB) interface via USB. We have connected 2 users on the RAN to assess MAC radio resource allocation between 2 users but not more to keep the BS stable.

#### 2.2.3.4 Air Interface

In this testbed, the Air interface is made ideal ($SNR > 12$ dB) and does not vary in time. Though such a channel is not present in a real-world RAN, this ensures a controlled air interface with a stable excellent channel quality rather than mobile users with uncontrolled radio interface phenomena. To do that, UEs are put in RF shield to protect them from outdoor interference and avoid multipath.

In this configuration, we cannot assess the ability of the RAN: 1) to recover air interface losses and errors; 2) to handover between cells; 3) to adapt modulation and coding scheme to air interface capacity variations. In the same time, it allows us to assess other aspects of RAN without to include in analysis unclear issues from air interface channel quality variations.

#### 2.2.3.5 RAN configurations

We use the Band 7 as defined by Evolved Universal Terrestrial Radio Access (E-UTRA) for air interface (see Table 2.2) as the 5G frequencies are not yet stable in OAI. Band 7 uses FDD with 10 MHz bandwidth (=50 PRBs) Single-Input Single-Output (SISO) at 2.68 GHz. Downlink modulation is up to 64-QAM and uplink modulation to 16-QAM. With such configuration, theoretical achievable throughput is about 37.5 Mbps (3GPP [39]-§7.1.7) in the downlink and 17 Mbps in the uplink (3GPP [39]-§8.6.2).

In addition, Appendix B.B.1 shows the RRC configuration sent by the mobile network core to the RAN at the establishment of the default DRB for PDCP, RLC, MAC and PHY layers. One bearer is established per user. At the RLC layer, the Acknowledged Mode (AM) is used, which ensures a better reliability with acknowledgements. UE's MAC

7. `https://www.gsmarena.com/leeco_le_2-8053.php`
8. `https://www.gsmarena.com/xiaomi_mi_10t_5g-10473.php`

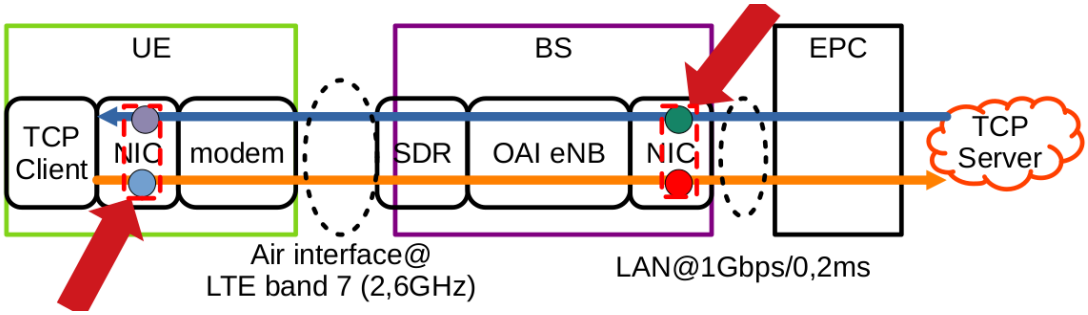



Figure 2.4 – The 2 passive capture points on the testbed pointed with red arrows.

layer Logical Channel prioritization (LCP) uses `logicalChannelConfig` configurations (*e.g. bucket size and prioritised bitrate*), but not used in the followings since only one default bearer is established per UE. At MAC layer, HARQ procedure is configured with 4 retransmission rounds (i.e. the UE will try 4 times to transmit a transport block or for $4 \times 8$ ms $= 32$ ms). Power Headroom (PHR) for the uplink radio transmission power is generated at least one time every 500 ms. Buffer Status Report (BSR) periodic timer is configured to 64 ms by default (in comparison with default 32 ms for srsLTE). `retxBSR-Timer` is introduced to improve BSR robustness by avoiding deadlock situation. 320 ms after a BSR without grant, UE will send another regular BSR. The Scheduling Request periodicity is set to 10 ms according to the SR config index. Discontinuous Reception (DRX) is not configured in the experiments.

#### 2.2.3.6 Measurements

A valuable contribution of this testbed architecture design is the capture at multiple points of the packets' timestamp (see Figure 2.4). Network captures are made at the UE, base station machine and server network interface. Network captures observe latency and throughput but also gives a more complete information of how network is transmitting data with a transmission pattern. Such data files are valuable for a deep analysis of latency-related RAN phenomena.

## 2.3 Traffic generation

A number of evaluation and validation scenarios are provided by International Telecommunication Union (ITU) and 3GPP. Scenarios [89]-6 describe the environment (e.g. urban or rural), users (e.g. stationary or pedestrian), deployment (e.g. sites density) and traffic

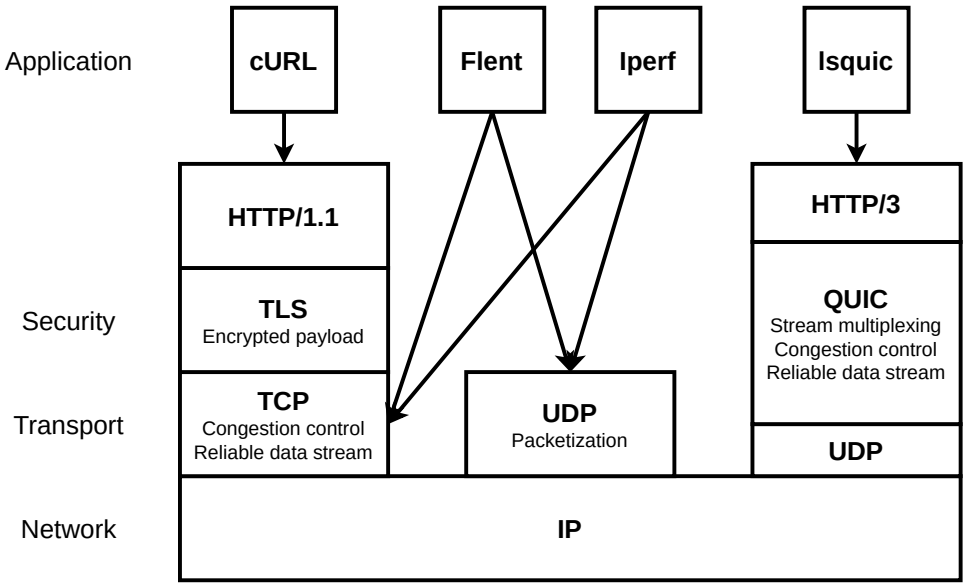



Figure 2.5 – Network protocols and applications in use on the testbed.

patterns. A survey on scenarios adopted is presented in [12], especially in section III.B, the 3GPP scenario parameters are described (i.e. what is a rural deployment or what is a FTP transfer). The testbed under this classification is an indoor hotspot-eMBB with 1 or 2 users per transmission without speed mobility. However, we do not use the traffic model proposed by the 3GPP (summarized in [12]-Table V). Traffic models are described with a file of size $S$ and a packet transmission distribution $D$. For instance, a FTP traffic (e.g. an TCP long transfer) is described by a truncated lognormal distribution with parameters $\sigma = 0.35$ and $\mu = 14.45$ that is an important simplification of the TCP transfer dynamic (the same arguments for web-browsing and video streaming).

We prefer to utilize linux internet stack to get representative results. Transport protocols and endpoint applications have been chosen to be representative of the common mobile networks usage by the customers. Transport protocol stacks (introduced in §1.2) and the applications are depicted in Figure 2.5. We also use a User Datagram Protocol (UDP) tool for the generation of a determined traffic profile, less for representativeness than to test the behavior of the RAN under certain conditions. This section presents the characteristics of the traffic generator used in the testbed.

**Iperf** Long TCP and UDP traffics are generated by iperf tool [180] installed on the server and on the UE. Iperf has a non-negligible advantage over an HTTP traffic generated for instance by watching a Youtube video that is the fine control on transmission

parameters of the experiment and a lean application layer. Watch video on internet or load webpage with chromium is factually more realistic to what experienced real user but with a loss on control on transmission parameters (e.g. datarate) or with undefined application behaviour (e.g. excess of buffering). Iperf is used to evaluate the transmission performances of the RAN for TCP and UDP long flow traffic. Especially, iperf TCP to test mechanisms related to TCP, congestion control and latency handling and TCP adaptation to RAN capacities. TCP protocol stack reference is the one of the *Linux kernel 4.15.0*[9]. Iperf UDP for traffic pattern generation with different controlled datarate and packet size.

With iperf, we are able to generate an UDP data flow with a determined datarate, packet size and jitter in the packet inter-departure. Uni-directionnal UDP flows are very useful to generate determined traffic profile that should not be representative of a real application behaviour but interesting the network response to the traffic profile i.e. how it will transport it.

**cURL** HTTP requests are performed with cURL. Compared to raw TCP downlink traffic, HTTPS requests have characteristics to be a short flow of few megabytes and for which the time to download is more important than the TCP goodput.

**lsquic** For the OAI testbed, we also set up a Quic client/server as Quic protocol. Pushed by Google, Quic will carry the most of the data in cellular networks in the upcoming years. Many android applications are using natively Quic protocol such as Youtube with Cronet [10]. Cronet is the chromium's networking stack. It implements Google Quic protocol and supports IETF Quic standard. However, in the same idea as Iperf compared to Chromium, we rather prefer to use a toy client/server for protocol testing. The objective is that to minimize undefined interactions with upper layers and focus testing on interaction between IETF Quic protocol and cellular network. We use lsquic [11] [181] toy client and server. We do not see elsewhere the use of lsquic on Android smartphone as client and server. The compilation and installation procedure are detailed in a note of blog [12]. The method proposed is an important contribution of our testbed design since it allows testing how Quic reacts to cellular network conditions. The toy server allows us

9. Linux TCP stack reference on the *kernel.org* repository ; `https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/net/ipv4/tcp.c?h=v4.15`
10. `https://developer.android.com/guide/topics/connectivity/cronet`
11. `https://lsquic.readthedocs.io/en/latest/`
12. `https://frj.medium.com/compile-an-ietf-quic-stack-for-android-and-test-it-on-a-cellular-radio-acc`

to generate Quic traffic and HTTP/3 [72] requests. lsquic implements Bottleneck Bandwidth and RTT (BBR) congestion control algorithm with a PTPC (Packet Tolerance PID Controller) method for the Acknowledgment (ACK) generation that is not representative of the majority of the quic stack.

**Flent** We complete the UDP and TCP using the Flent [182] tool that is today usually used in the bufferbloat and transport protocol community to run tests on real network hardware. It is based on netperf tool (an alternative to iperf developed by HP) and provides a helpful software overlay with scenarios and visualizations to increase reliability and making easier network experiments. At the best of our knowledge, Flent was never used in an RAN testbed. The difficulty was to adapt and compile netperf server for Android and Flent.

## 2.4 Analysis tools and methodologies

In this section is presented the methods used to determine latency and impact of radio segment on traffic flow. It concludes the chapter concerning research method after a complete review of testbeds and traffic generation.

### 2.4.1 Active RTT measurement

A common method to get end-to-end latency experienced by a user in a network is by measuring simultaneously to a traffic flow, the RTT with an Internet Control Message Protocol (ICMP) ping traffic. A ping request is sent to a destination host and in response the host respond with an echo packet. Sent regularly, the ping samples actively the RTT of the network. ICMP ping does not inform about the composition of the latency by segment, e.g. uplink/downlink or RAN/core. Moreover, ping takes "snapshot" RTT at a specified frequency leading to a sampling issue. It does not inform about traffic transmission pattern.

In parallel of that, IETF standardized with RFC5357 [183] a Two-Way Active Measurement Protocol (TWAMP). TWAMP has better accuracy to measure latency, jitter and packet loss than ping, however it reveals to be more complex to be put in place in the mobile environment.

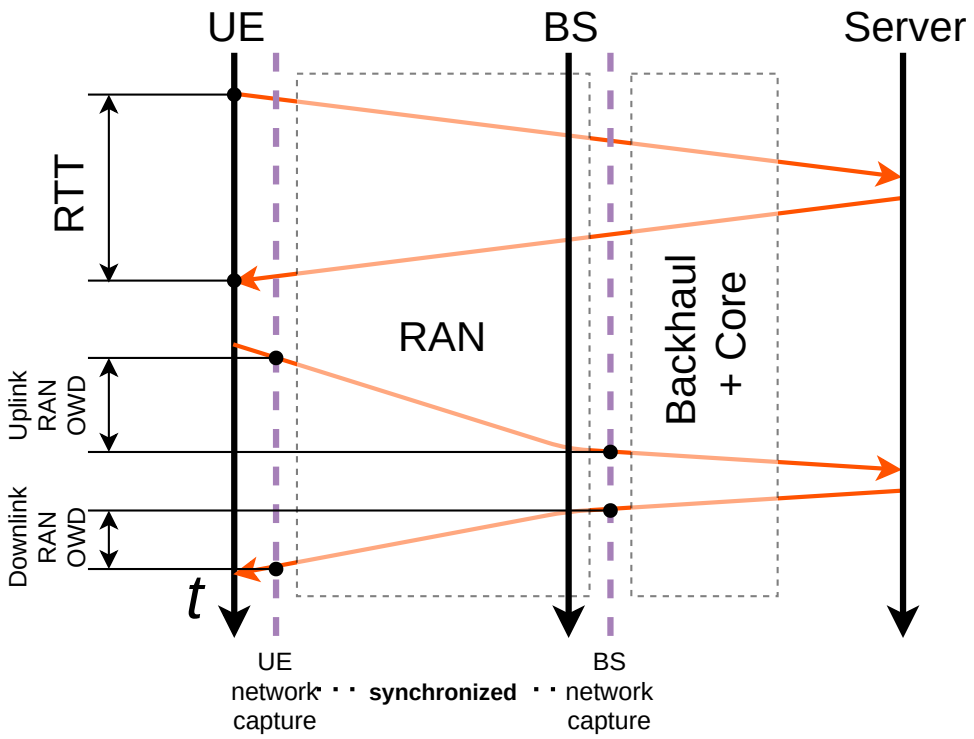



Figure 2.6 – 2-points passive capture latency measurement.

### 2.4.2 Passive measurements

The technique of passive measurements [184] consists of taking network captures at the edge of the RAN. "Passive" because the capture simply observes packets on network without interact with them or with the network. [185] employs this technique to analyse the RTT and latency variation of TCP connections. The method illustrated in Figure 2.6 uses 2 measurement points at which timestamped packets are copied in a network capture. The 2 points of measurements should be at the edges of the RAN. Practically, measurement point is a *tcpdump* capture at the Operating System (OS) cellular network interface. In the UE, the network capture is made at the network interface mounted by Android when the modem is enabled. Then, captures do not include cellular network protocol stack. The other capture is made at the BS's interface connected to the backhaul and the core, captures include all the cellular network stack. Thanks to the 2-point measurements, a packet is observed at the entry and the exit of the RAN. The RTT is computed from one capture by observing the departure of a packet and the response corresponding to it without adding active measurement packet in the network. The uplink delay is computed from 2 captures, as the time for a packet departing from UE to reach the output of the

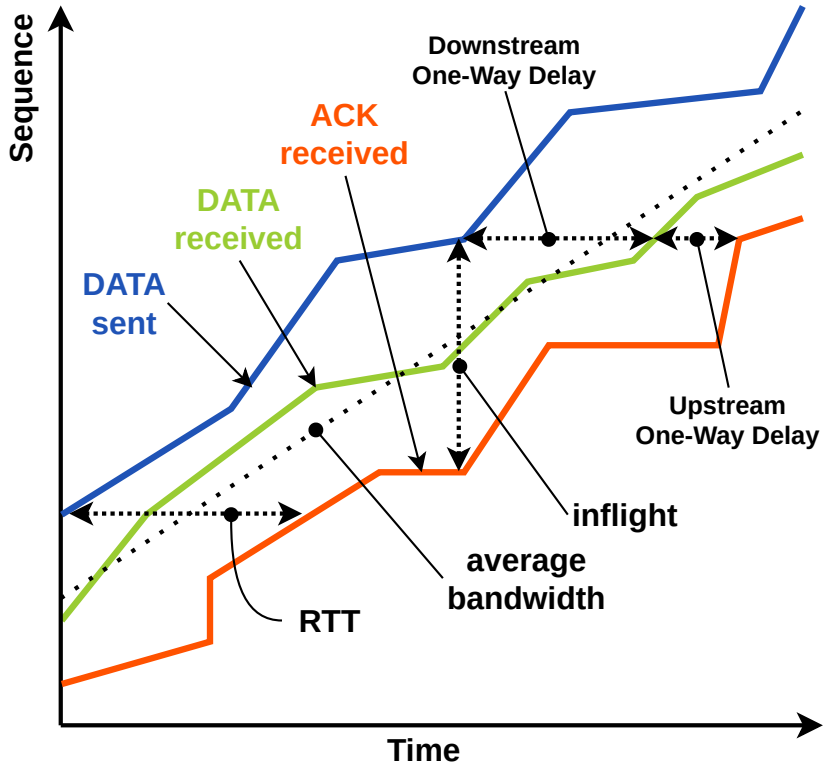



Figure 2.7 – TCP time-sequence plot reading.

RAN. Conversely, the downlink delay is the time for a packet from the arrival on BS machine to reach UE network interface. Packets are uniquely identified with their IP identifier included in the IP header. The measurement of network performances is then known at a packet-level.

Moreover, network captures give a very useful information on the manner data are transmitted through the RAN with the *traffic profile* plotted in Figure 2.7. The traffic profile is the cumulative volume of the transmitted bytes over time. The traffic profile of a continuous transmission will lookalike a line. Bursty transmissions will have a stepped traffic profile.

Traffic profile includes a lot of information that the simple measurement of the delays does not have. For instance, the increase of the delay could reveal the build up of a temporary queue due to a lack of transmission opportunity, or it could be the sign of a too important transmission datarate compared to network capacities, leading also to the build up of a queue. In the first case the traffic profile shows emphasized steps while in the second case the traffic profile shows a transmission as a line with a slope inferior to

the generation datarate. That is significant in the following of this work to quantify the source of latency and jitter among application transmission pattern, network capacities, retransmissions, resource allocation scheme. We think that is an aspect too often put apart in work concerning latency in wireless networks.

Network calculus [186] mathematical method is largely based on the concept of the traffic profile and consider each segment of the network as a node with internal characteristics which modify a traffic profile in input to a traffic profile in output. A convenient representation of traffic pattern at 3 capture points is given in Figure 2.7. The horizontal difference gives for selected bytes the time it spends in the network. The vertical difference gives at the selected time the amount of bytes in the network. In our case, the vertical difference is limited by the buffers' length in the base station and in the UE modem. The maximal slope of the line orange is limited by the network instantaneous throughput (in the downlink or uplink).

However, the task of synchronizing the 3 captures made with 2 different clocks at *sub-ms* level is challenging. It is challenging especially because of this need to have a clock synchronization below the radio slot level.

### 2.4.3 Network captures synchronization

The 2 machines on which are realized the captures are quite a bit different. We could not ensure to have a *sub-ms* synchronization between CPU clocks of UE and BS over a long timescale. The solution of Network Time Protocol (NTP) server is not suitable because it does not warrant such degree of synchronization. More generally, network-based synchronization does not have a sufficient degree of precision with about $2-3$ ms. The solution found in the cellular networks to synchronize nodes, especially base station is usually to use a GPS signal. But for a sake of simplicity and for the cost, we opt to a solution with the measurement of the timestamp on both UE and BS at the beginning of the experiment and at the end. This approach is based on the assumption of that the clock drift during a seconds-experiment is contained to lower than one millisecond.

As a reminder, the UEs are connected to the BS machine through ADB interface (i.e. USB). We tried a first approach by which in parallel we get the current timestamp of the host machine and the timestamp of the UE using ADB remote shell. After evaluation, we found a standard deviation of 4.5 ms that is too important to have a good synchronization of captures. Root causes are multiple, among ADB server request, the USB interface and android shell and kernel delays, we could not evaluate which one is in cause of this variation

of the measurement.

The solution we finally opt for is as follows: 1) Mount UE as a tethering server at which the host connect to; 2) Capture packets on the UE and on host tethering network interfaces; 3) Send 5 pings packet through USB interface that will be captured at the both endpoints; 4) Stop capture and unmount tethering; 5) For all pings in the 2 captures, compute the timestamp difference and uncertainty due to USB interface; 6) Return the mean timestamp difference in second between the 2 endpoints and return the uncertainty.

After evaluation, we found a standard deviation of about 0.35 ms over 5 seconds that is below 1 ms. Before an experiment, the `androsync` script is run and returns a timestamp offset between the 2 network capture points. We ensure that for a ten seconds transmission, the time drift between the 2 capture points is below the radio subframe. We consider this precision sufficient to pursue radio access network latency studies.

## 2.5 Conclusion

In our study, we are interested in the latency performance of RAN segment in controlled good radio conditions. The testbed presented in section 2.2.3 is designed to focus on radio link layer. We selected the open-source project OpenAirInterface as LTE RAN solution after a short review in Sec. 2.2.1 of the existing RAN solutions. The BS is connected to an OAI EPC for the server and an USRP SDR card for the RF transmission. 2 COTS UEs are connected to the LTE cellular network with a default BE bearer.

Testbed main limitation is related to the air interface model that is too simplified to be realistic. We can add to limitations:

— No handover procedure linked to a mobility event
— The lack of Remote Radio Head (RRH) and antenna segment
— A long development process to implement a feature by the complexity of the code (1.13M Source Lines of Code (SLOC)).

We reviewed in section 2.3 the traffic generation method. A strong focus was put on transport layer protocol, more than on a limited number of targeted service. We justify this choice by the perspective of generalization we would like to give to this work in a sense where we think that a major part of services will be based on UDP/TCP/Quic protocols. To avoid undefined behaviours, implementations and configurations issues from higher level protocol, we place our study at the layer just above IP. The RAN should guarantee the best transmission as possible in terms of throughput, latency and jitter to end-to-end protocol. For that, TCP and Quic servers are simple as possible to avoid hypothetical higher layer issues. To evaluate radio capacity and traffic pattern transmission, we use UDP with a variable datarate and packet size.

Finally, section 2.4 presents the latency calculation method with both active and passive measurement. As we show, two network capture points are sufficient to have an information at the packet-level of the total RAN latency on the condition of having a synchronization of the clocks of the 2 captures.

The next step is to obtain such fine knowledge of the latencies inside the RAN i.e. inside the BS node. The BS is usually a black box which pulls latency metrics for instance per layer or per QoS bearer. In the following chapter we present a novel method to measure at packet-level internal latencies inside the base station.

Chapter 3

# LatSeq

**Abstract** : *This chapter presents a major contribution of this thesis that is a novel measurement tool, called LatSeq, for Radio Access Network (RAN) internal latency analysis. After a review of motivation behind this work, we introduce the notions of internal latency, packet fingerprint and packet journey that are used for the RAN latency analysis. We particularly detail the key design of the tool to achieve low impact on the observed system and the relevant fine-grain latency statistics. The evaluation with a showcasing of the tool demonstrates these properties. Finally, we discuss the importance of the contribution for the open source community.*

## Contents

**3.1 Motivation . . . . . 63**
**3.2 Definitions . . . . . 64**
3.2.1 Internal latency . . . . . 64
3.2.2 Packet fingerprint . . . . . 66
3.2.3 Packet journey . . . . . 68
**3.3 Implementation . . . . . 70**
3.3.1 Real-time measurement of packet fingerprint . . . . . 71
3.3.2 Measurement points in the LTE stack . . . . . 76
3.3.3 Internal latency analysis . . . . . 79
**3.4 Evaluation . . . . . 84**
3.4.1 Unitary measurement point generation time . . . . . 85
3.4.2 Memory usage . . . . . 86
3.4.3 In-running global performance . . . . . 86
**3.5 Related work . . . . . 88**
**3.6 Showcasing LatSeq: TCP ACK packet bursting in uplink . . 89**
**3.7 Contribution to open source community . . . . . 91**
**3.8 Conclusion . . . . . 92**

**Publications**

— *LatSeq: A Low-Impact Internal Latency Measurement Tool for OpenAir-Interface.* **Flavien Ronteix–Jacquet**, Xavier Lagrange, Isabelle Hamchaoui, Alexandre Ferrieux, Stéphane Tuffin. IEEE WCNC'21, March 2021, Nanjing / Virtual, China. DOI : `https://doi.org/10.1109/WCNC49053.2021.9417345`, Link to paper : `https://hal.archives-ouvertes.fr/hal-03244279`

— Source code of LatSeq tools in OpenSource : `https://github.com/Orange-OpenSource/LatSeq`

— Official OpenAirInterface repo of the LatSeq implementation in OAI LTE base station : `https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/latseq`

## 3.1 Motivation

Offering low latency communications is undoubtedly one of the foremost 5G promises. However, coping with latency objectives under few milliseconds creates tough challenges for the air interface design but also for QoS management and resource allocation on the radio segment. In general low latency mechanisms proposed in the literature were only evaluated in simulated environments. Several open-source platforms of end-to-end mobile networks have recently emerged: OAI and srsLTE are oriented toward research purposes. Both open-source and commercial Base Station (BS)s report a bunch of metrics to evaluate the quality of the network and possible issues on Key Performance Indicator (KPI). Usually, latency is given per layer or per QoS bearer from a sampling process. In OAI many timers are set up to measure the contribution of different network functions to the total latency. We quickly understand the limit of this approach to pursue a comprehensive evaluation of the latency in the RAN. We would like in one hand to correlate delays reported by the BS with the latency measured with the 2-point passive captures technique and in the other hand, to get composition of delays for an individual packet or at least to a packet flow.

A comprehensive tool for latency analysis added in software BSs is needed and constitutes an important contribution of this thesis. The tool should be able to measure the delays in each layer entity but also delay values related to the same packet should not be captured independently of each other: what should be built is the sequence of successive latencies across the layers. Making measurements inside a running BS is a challenge in itself as it requires measuring delays of the order of a microsecond for thousands of high throughput connections and ten of diverse users. Logging these large numbers of events necessarily incurs extra delays, while we are precisely seeking a low-impact tool to preserve the BS operation. To meet these double harsh constraints of fine granularity and performance we develop a tool called **LatSeq** — "*Latency Sequence analysis*". The principle behind LatSeq could be summarized as follows: During in-running BS, solely log packets' information needed to compute in offline packet's sequence of latencies. From this principle comes the design of the tool presented in section 3.3. Before that in the next section 3.2.1 we define the notions of internal latencies and sequence of latencies extensively used in this work.

## 3.2 Definitions

### 3.2.1 Internal latency

The *end-end-end latency* experienced by a packet is divided between heterogeneous segment by which this packet is carried (see Figure 1.8). The three fundamental sources of latency in computer networks is the propagation delay, the buffering and the processing delay. What affects the QoS is the end-to-end latency in light green in Figure 1.8. It aggregates all the types of latency accumulating on the packet's path. But latency characteristic is radically different between an international fiber and a wireless access segment. The question is, in the RAN, what is the type of latencies, their magnitude and their location. With the setup presented in section 2.2, the cellular network latency per packet is known accurately, but it tells nothing on RAN internal latencies in blue. The solution we propose complete the study of end-to-end latency in red in Figure 1.8 with a focus on base station internal latency in green.

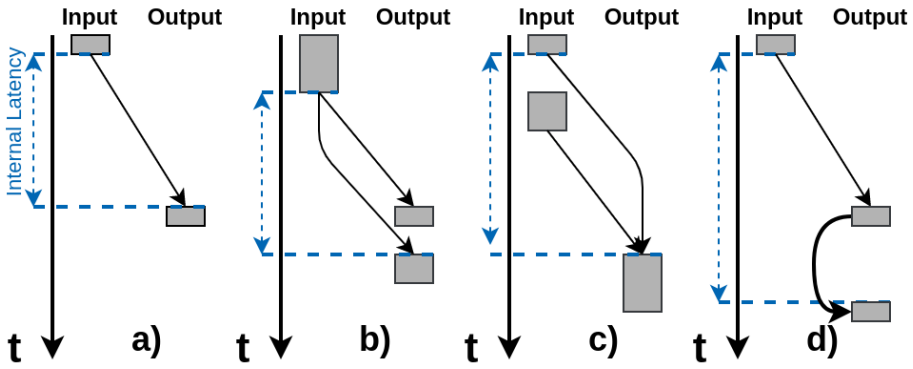
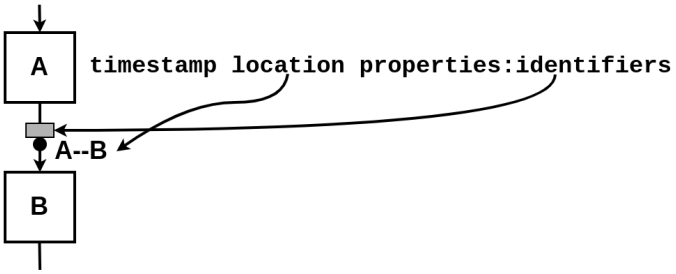



Figure 3.1 – Internal latency cases. *a) Linear; b) Segmentation; c) Concatenation; d) Retransmission*

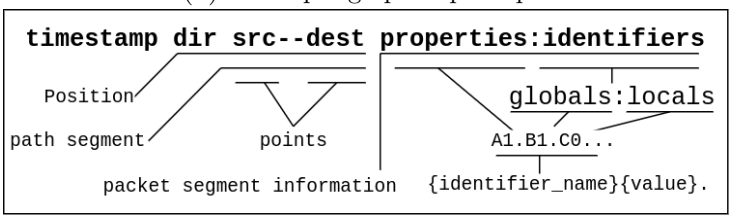
Defining the latency of a packet handled through the radio access network is in itself difficult. A packet is processed by a collection of layer entities and is prone to segmentation or reassembly, concatenation and retransmission. This induces 4 cases for internal latency as shown in Fig. 3.1. The latency can be considered either for each packet segment or for all segments related to the same packet. Moreover, depending on the point of view (from inside the node, from outside the node, from software, from hardware), delays are not of the same nature and not always measurable. We define the **internal latency** as the time between the moment when the packet is fully received by the node (here the base station) from the input interface and the moment when all the segments making up the packet

leave the software part of this node, to be transmitted and possibly retransmitted. The node is a logical environment with a unified reference clock and delimited by an input and an output physical interface. The definition catches the latency effectively experienced by the packet due to software part of the node. The different cases we encounter in the cellular network node are illustrated in 3.1. For a packet that is buffered and processed by the node as a whole (case (a)), the internal latency represented by blue arrow is the difference of time between the arrival and the departure of the packet. The worst case corresponds to the case (d) when a multiple retransmission occurs, delaying the effective correct transmission of the packet. A lost packet is considered to have an infinite internal latency. This definition of internal latency is particularly suitable for softwarized networks where network functions are implemented as a software that is increasingly the case with the development of cloud RAN and network function virtualization. In the same time, this definition of internal latency does not include latencies due to access or physical layer.

A second challenge is to track a packet or a data segment along its journey inside the node. In the following, we call a packet or a data segment in any layer *a data unit*. The node consists in a succession of layers and functions that process and buffer packet. Layers entity is a coherent environment that provide a network service. For instance, the MAC layer ensures the (de-)multiplexing of packet for physical layer and uses a buffer to perform retransmissions. The MAC entity is associated to a baseband unit. Each layer manages a set of local identifiers to differentiate entities. Local identifiers are used to route packets through layers. At the bound of two layers, local identifiers link two entities. Inside the entity, a common manner to differentiate packets and track through entities is to assign them a temporary identifier. For example, each UE is identified at the MAC layer by the Radio Network Temporary Identifier (RNTI). Also, several flows can be transmitted to the same UE by use of different DRBs. At the RLC level, each data unit is referenced by a Sequence Number (SN) and multiple logical channels can be multiplexed and identified by a Logical Channel IDentifier (LCID). Then, a given packet with a SN number will be routed through MAC and RLC layer using DRB identifier, RNTI and LCID. Aggregating the identifiers from different layers makes it possible to track a packet along its path. Note that some identifiers can be redundant (e.g. the DRB number and the DRB) but this does not present a problem.




(a) LatSeq fingerprint principle




(b) LatSeq fingerprint format

Figure 3.2 – LatSeq fingerprint.

### 3.2.2 Packet fingerprint

It comes what we define the packet fingerprint at different layers. The packet fingerprint is generated whenever a measurement is made to track a packet segment during its life within the node. It includes the identification of the packet with the mentioned local identifiers but also the position of this packet in the node and a precise timestamp as illustrated in Fig. 3.2a. The syntax of the packet fingerprint is given in Figure 3.2b. The main elements are the following ones:

1. The *timestamp* with a precision of at least one microsecond, largely below the length of an OFDM symbol of 1/14 ms;
2. The *location* of the packet on the path indicates the direction, (i.e. downlink or uplink) followed by the network segment observed;
3. The packet fingerprint *local identifier* part is composed of a set of packet-related or layer-related identifiers with the name of the identifier followed by its value.

The packet fingerprint does not include all the local identifiers accessible at a layer and does not include packet header or packet payload at all.

We remark in the case of radio access network that there are also *global identifiers* that are common for a packet flow. That is the case of the RNTI that is related to a unique user session and accessible from entities' data-structure almost all along the path

from high PHY layer to PDCP. It is convenient to add global identifiers for a question of filtering.

In addition to latency measurement, that is the primary objective of this tool, we added another set of values that is related to *packet properties* not used to identify a packet. In computer networks, it is basically the size of the packets, but also information related to the radio context of the packet, for instance the buffer occupancy at the arrival of the packet in the buffer. The cost for the packet fingerprint captures is limited and provides valuable data for latency analysis.

The real-time capture of fingerprints in-running BS poses a difficult engineering challenge. The main advantage of the proposed approach is to limit the quantity of information that has to be captured of a packet to retrieve packet path and latencies. Contrary to regular passive measurements that capture by copying bytes of entire (or simply the header with optimizations) packet, we copy only values of related packet and entity data structures that are useful to identify a packet at a position.

```
1 634.770255  U  mac.demux—rlc.rx.am  len107:rnti22805:drb1.lcid3.rsn36.rso0
2 634.770257  U  rlc.rx.am—rlc.reas.am  len107:rnti22805:drb1.rsn36.rso0
3 634.770360  U  rlc.reas.am—pdcp.rx  len163:rnti22805:drb1.rsn36.psn29
```

Listing 3.1 – Consecutive LatSeq fingerprint of a packet segment

An individual packet fingerprint in fact cannot be used as is to compute latency, two are necessaries to capture instant latency at a segment. The listing above shows 3 fingerprints from a capture. The fingerprint 1. indicates that the packet with RLC SN 36 has been measured at the bound of RLC and MAC layers at second 634.770255. The destination of the packet at this fingerprint is the `rlc.rx.am` point. The second fingerprint 2. source's point is `rlc.rx.am`. Because the DRB identifier and the RLC SN are in common in fingerprints 1. and 2., then they correspond to the same packet but at 2 different layers and at 2 different moments. The latency experienced by this packet at the `rlc.rx.am` is then of 2 $\mu$s. A similar analysis leads to conclude that the third fingerprint belongs to the same packet at the bound of the RLC and the PDCP layer with a delay of 103 $\mu$s introduced by the `rlc.rx.am` segment. Thus, common local identifiers and matching source/destination fields reflect the continuity of the packet's life across layers.

### 3.2.3 Packet journey

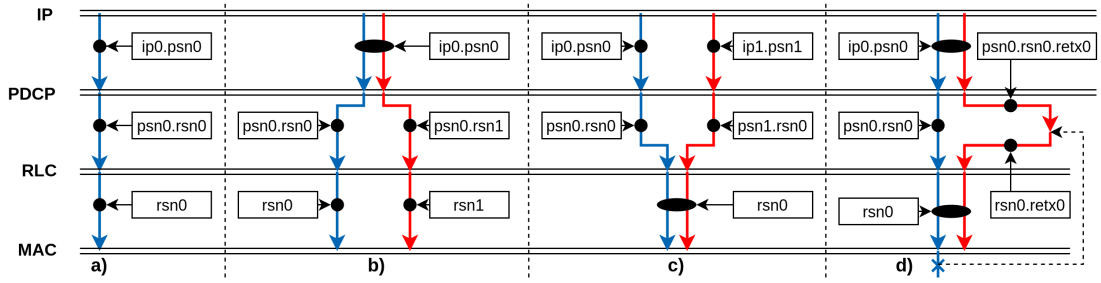



Figure 3.3 – LatSeq journeys cases. *a) Linear; b) Segmentation; c) Concatenation; d) Retransmission*

We call the packet's life across layers the **packet journey** as the list of successive fingerprints for a given packet. Journeys of packet are mixed together in the trace file generated during the execution of the BS. In Figure 3.3, fingerprints represented by rectangles, the packet path is `IP-PDCP-RLC-MAC` but due to segmentation and concatenation, a given fingerprint can be included in several journeys. Conversely, fingerprints related to different journeys are interleaved in the trace file. As a consequence, journeys are not distinguishable at first glance and should be rebuilt using appropriate processing.

In a way, journeys are **an oriented graph**. Each fingerprint can be seen as a vertex of a graph. When the destination field of a fingerprint is the same as the source of another one, there is an edge between the two vertices if both fingerprints share the same local identifiers' value. We are thus defining subgraph each one corresponding to a packet journey. These subgraphs are Directed and Acyclic (DAG) [187]. Such graphs are well-used to represent a network of processing elements. All fingerprints generated at a given input share the same source field. For example, the fingerprints of all downlink packets arriving at the BS on the wired interface are characterized by `src=ip.in`. Similarly, when a data unit is transmitted to the radio interface, the destination field of all its possible leaf fingerprints are `dest=phy.out`. The 4 typical graphs possible in the network node are illustrated in Figure 3.3. When a data unit is dropped (or loss) by a layer, a new journey is generated replacing the previous one as shown in the case d).

From the principles of packet fingerprints and packet journeys, we build the *matrix of internal latency* (Fig 3.4). The vertical axis corresponds to the timestamp, and the horizontal axis represents the packet segment path. The fingerprints (blue points) are put on the grid according to timestamp and packet segment path. Coherent blue points related to a packet journey are linked together with the blue line. The total internal latency

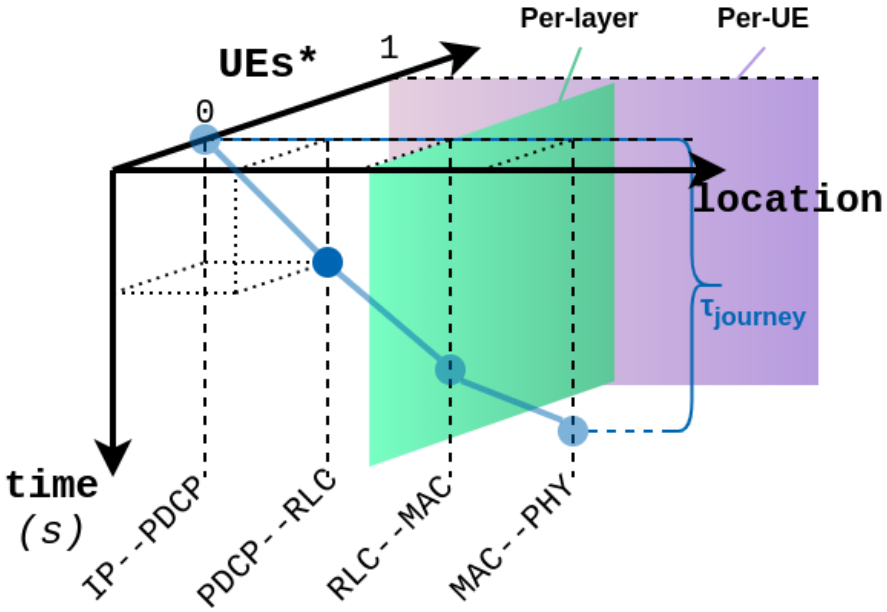



Figure 3.4 – Matrix of internal latency. *One axis for the time, one for the location and one for a filtering parameter, e.g. UE.*

experienced by a packet (or a packet segment) in the BS corresponds to the difference of time between the incoming and the outgoing edges of the journey graph. Different internal latencies are calculated with the timestamp value assigned to the vertices of the graph. In addition to the latency related to an individual packet, our approach is able to provide latency statistics per packet path and per layer. The third axis is used as a free parameter for filtering. For instance on the figure the third axis is used to split packet journeys per UE. All the information related to the internal latency in the BS is hold in the matrix with each point corresponding to a packet fingerprint.

## 3.3 Implementation

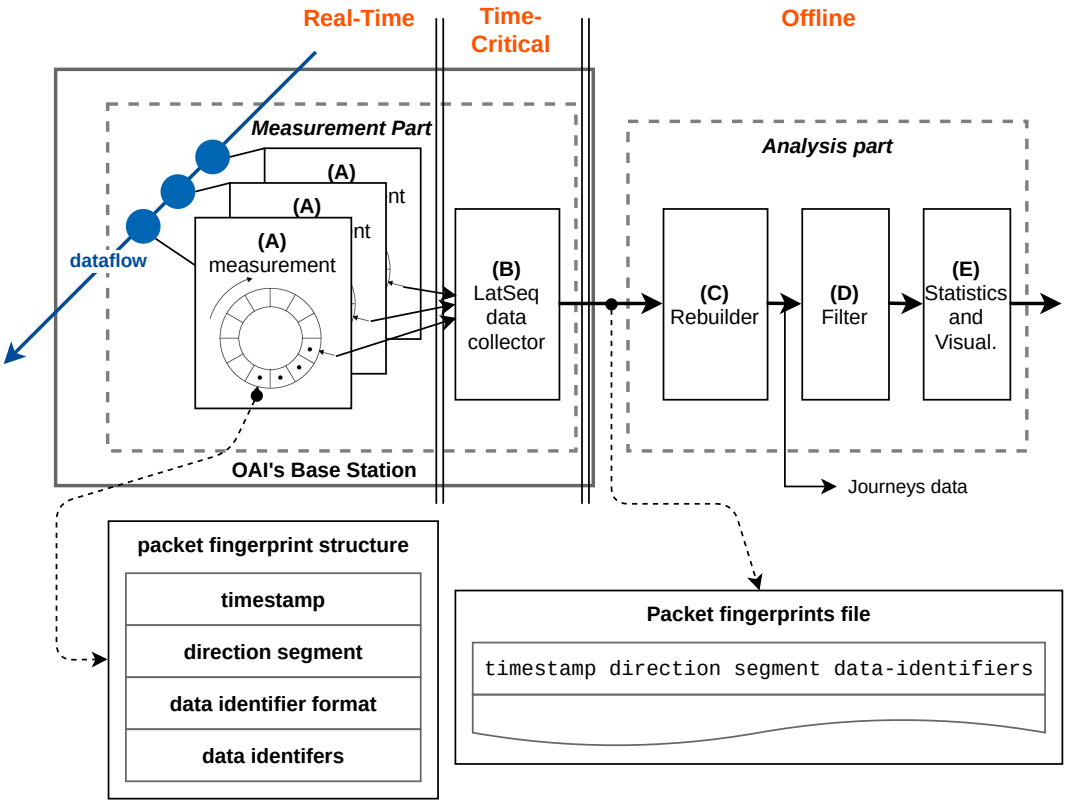



Figure 3.5 – LatSeq architecture.

We have implemented our proposed internal latency measurement tool for the OAI RAN project. LatSeq is composed of two functional blocks as shown in Figure 3.5 that corresponds to the 2 challenges we have previously taken up, the real-time measurement and the packet journey rebuilding. The real-time capture of fingerprints is associated to the measurement part. The packet journey rebuilding and the computation of internal latency is associated to the analysis part. Another practical hard work comes when we implement the tool in an existing project, which is the placement of the relevant measurement points. In this section we detail the rationales, the implementation of LatSeq and the integration with OAI eNB code.

| Mechanism | U | C |
|---|---|---|
| Variadic macro | ✓ | |
| Inlined measurement function | ✓ | |
| Time in number of CPU cycles (rdtsc) | ✓ | |
| Thread Local Storage (TLS) | ✓ | ✓ |
| Lockless ring buffer for temporary local storage | | ✓ |
| Formatting by the data collector thread | ✓ | |
| Buffered IO | ✓ | |

Table 3.1 – Implementation choices for LatSeq measurement module. *(U: low CPU Usage, C: low CPU Contention)*

### 3.3.1 Real-time measurement of packet fingerprint

The measurement part (Fig. 3.5 (A) and (B)) captures fingerprints on a running BS and records them in a trace file. Thus, we have measurement points on the packet path that necessary introduce a processing overhead to the BS. The design of the measurement part focuses on reducing the processing overhead introduced by the measurement in itself. We reduce the overhead by minimizing the number of tasks to perform on the packet path and by isolating time and IO consuming tasks (i.e. writing to the trace file) from the rest of the BS program. It exists a separation between the measurement and the conversion stage (see Appendix C.C.1 where the LatSeq main thread runs in parallel to the OAI threads where measurements are made). Such design isolates the time and IO-consuming task of fingerprint generation from the rest of the program where measurements are performed. The strategies and mechanisms utilized to achieve low-impact challenge are summarized in the Table 3.1 and developed in the following.

#### 3.3.1.1 Measurement points

The point of the electron microscope (Fig. 3.5-(A)) is a macro-instruction called `LATSEQ_P`. It is inserted in the code with one `LATSEQ_P` corresponding to one **measurement point**. The format of this macro-instruction is `LATSEQ_P("dir src-dest", "data_id1%d...", data_id1, ...)`. For instance the code in Listing 3.2 associated to the latseq measurement point at the `rlc.tx.am-rlc.seg.am` segment:

```
#ifdef LATSEQ
LATSEQ_P("D rlc.tx.am--rlc.seg.am",
```

```
            "len%d:rnti%d:drb%d.sdu%d.rsn%d.rso%d",
            sdu->size, entity->ue_rnti, entity->channel_id, sdu->sdu_num, pdu
                ->sn, pdu->so);
#endif
```

Listing 3.2 – measurement point code

At the compilation of the program, the macro-instruction is replaced by its inlined form into the OAI code. The inlined form avoid a costly function call operation. There are as many times the `LATSEQ_P` code as there are measurement points on the packet path justifying the idea to drastically reduce the extra-delays associated to the measurement process.

The arguments contain all the necessary information for the fingerprint to be formatted in the trace file. Because the macro-instruction handles a variable number of data identifiers, we use the *variadic macro* mechanism. Variadic functions are functions that can take a variable number of arguments. Variadic function consists of at least one fixed variable and then an ellipsis (i.e. `"f(a, ...)"`) as last parameter. After a quick evaluation of the assembly code generated, the *variadic macro-instruction* reveals to be the most efficient in terms of instruction number compared to variadic functions. In fact, the variadic function needs a function call instruction (that costs instructions) and a call method to get access to arguments. Interesting properties of variadic functions are the readability of the code and the number of source line of code compared to variadic macro-instructions. For measurement points the performance has been preferred over code length and readability. Practically, when a `LATSEQ_P` is put with 3 arguments (the segment string, the identifier string and one data identifier), `LATSEQ_P("p", "f%d", i1)` is converted during the compilation time (and not at the execution as with variadic function) by `LATSEQ_P3` that is the macro-instruction for the inlined function `log_measure1(p, f, i1)`.

The `log_measure(p,f,...)` performs 2 tasks, takes a timestamp and copies arguments value into a data structure. The *timestamp* presents in the fingerprint is not given in argument but taken at the `LATSEQ_P` measurement. The CPU timing information in x86 architecture is obtained by using the `rdtsc` assembly instruction (the cost is one assembly instruction or few CPU cycles) and returns the CPU counter in CPU cycle since the boot (not a time in second). The timestamp counter (TSC) is a hardware counter in x86 CPU, therefore low-overhead and high resolution compared to commonly used `gettimeofday()` function in `C`. The `rdtsc` value in CPU cycle is not converted at the stage but directly used as timestamp for fingerprint. To get timestamp in second with a

microsecond precision, the CPU frequency has to be known precisely such as:

$$t_i = t_0 + (\text{rdtsc}_i - \text{rdtsc}_0)/F \tag{3.1}$$

where $\text{rdtsc}_0$ is the CPU cycle at the arbitrary start of LatSeq with corresponding UNIX time $t_0$ and $F$ is the CPU frequency in cycle per second (of order of GHz). The CPU frequency is not *a priori* known with precision and subject to variation over time *e.g. quartz drift or internal power mode management*. Also, the flag `tsc_constant` should be set on multicore CPU to ensure a cycle counter synchronization between cores on which are executing different threads of the program. To keep the precision of `rdtsc` at the conversion stage to UNIX timestamp, we log regularly (i.e. every second) a synchronization fingerprint with the format:

```
rdtsc_value  S  rdtsc—gettimeofday  gettimeofday_ns
```

Listing 3.3 – rdtsc synchronization fingerprint

During the fingerprints' generation, the CPU frequency drift is corrected with synchronisation point and the timestamp value is converted to UNIX timestamp. Our method enables a precision of the *sub-microsecond* precision with a limited cost during the runtime as the conversion is performed on the packet fingerprint file.

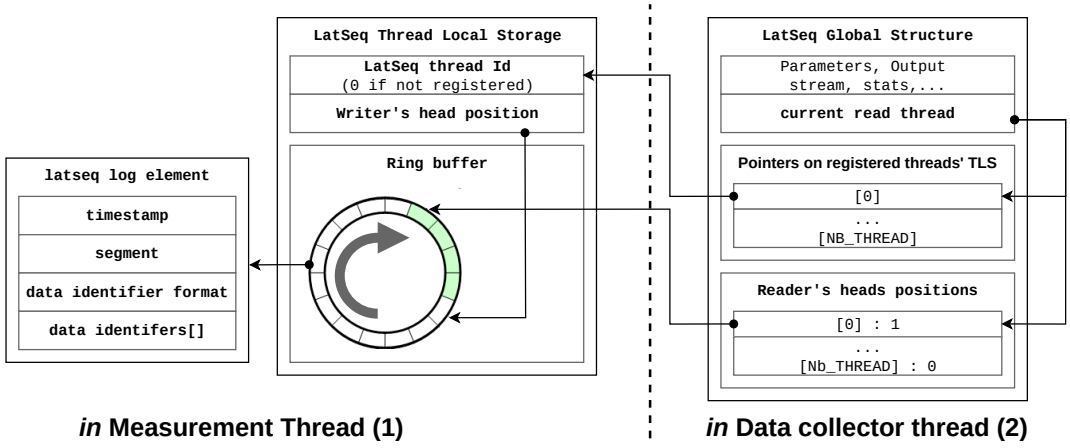



Figure 3.6 – LatSeq fingerprint data structures.

Concerning the arguments, the macro-instruction code puts the strings and arguments in a `latseq_element` data-structure presented in Figure 3.6. The *LatSeq structure* contains the timestamp, the string constant (relative to a measurement point) of packet path segment, the string constant for the identifier format (relative to a measurement point)

and a fixed-size array of arguments (to avoid time-consuming allocation and de-allocation of dynamic array). The arguments are simply copies in the LatSeq element with a low instruction cost. As soon as the *latseq_element* is filled, the OAI program is resumed.

In total, the decompilation of the `log_measure` function with 2 data identifiers gives 37 `x86` CPU instructions to perform a fingerprint capture (See Appendix C.C.2). Moreover, the instructions to get the LatSeq element could be optimized at the run time with a pre-fetch. Each data identifier added costs 3 instructions:

```
mov -0x8(%rbp), %rax
mov -0x24(%rbp), %edx
mov %edx, 0x1c(%rax)
```

Listing 3.4 – Assembly code to copy a data identifier

On a GHz CPU, each measurement point costs about **twenty nanoseconds**. For 10 measurement points, we delay a packet far less than a microsecond for measurements.

#### 3.3.1.2 Fingerprints collector and conversion

The *conversion stage* of latseq_element into a character string fingerprint is centralized in a dedicated thread. latseq_element data-structure are stored locally before to be collected and converted by the dedicated thread. As illustrated with the LatSeq data-structure Figure 3.6, the latseq_element is temporarily stored in a lockless ring buffer. The principle is as follows : As the packets are observed, fingerprints are put in the ring buffer at the writer head place. In the same time, the reader converts fingerprints that are behind the writer head. No fingerprints are lost if the reader is faster than the writer in average. The buffer size is here to handle an instantaneous burst of fingerprints production or a pause in the conversion. The buffer is said lockless because the writer and the reader can interact with the buffer without the usage of a mutex. A mutex is usually used to synchronize reading/writing of a variable by different threads. The ring buffer and the two heads, one to read and the other to write avoid the need of a mutex, mutex that could lead to unnecessary lock of the program. This buffer uses static memory that is local to the instrumented thread. It uses the *Thread-Local Storage* is a local memory to a thread instantiated at the initialization of the thread. In other words, there are as many ring buffers as the number of instrumented threads (not the number of latseq point as instrumented thread could have many latseq points).

Another solution is to write fingerprints in a non-thread local buffer. Concurrent

```
1612466634.770246 U phy.in.proc--mac.harq.up len565::ue0.cbseg2.fm237.subfm6
1612466634.770247 I mac.harq.up nack0:ue0:harq0.fm237.subfm6
1612466634.770253 I phy.srs ucqi163:ru0.ue0:
1612466634.770254 I mac.ind bsr0.len0:ue0:lcgid0
1612466634.770254 I mac.ind bsr0.len0:ue0:lcgid1
1612466634.770254 I mac.ind bsr0.len0:ue0:lcgid2
1612466634.770254 I mac.ind bsr0.len0:ue0:lcgid3
1612466634.770254 U mac.harq.up--mac.demux len107:rnti22805:ue0.lcid3.fm237.subfm6
1612466634.770255 I rlc.rxbuf.am occ107:rnti22805:drb1
1612466634.770255 U mac.demux--rlc.rx.am len107:rnti22805:drb1.lcid3.rsn36.rso0.rfi2.fm237
1612466634.770255 U rlc.rx.am--rlc.reas.am len107:rnti22805:drb1.rsn36.rso0.rfi2
1612466634.770256 U rlc.reas.am--pdcp.rx len163:rnti22805:drb1.rsn36.rso0.psn0.psn29
1612466634.770260 U pdcp.rx--gtp.out len163:rnti22805:drb1.psn29
1612466634.770265 U gtp.out--ip.out len161:rnti22805:drb1.psn29.teid5.gsn30.ipid0x5a4a
1612466634.770276 I mac.sched.down mcs28.tbs7.nrb3:rnti22805:lcid3
1612466634.770277 D mac.mux--mac.txreq len7:rnti22805:lcid3.txreq0.reqfm238.harq4.sfn3808
1612466634.770279 D mac.txreq--mac.harq.down len7:rnti22805:txreq0.harq4.sfn3808
1612466634.770310 D mac.harq.down--phy.out.proc len7:rnti22805:harq4.fm238.subfm0
1612466634.770370 D phy.out.proc--phy.out.ant len15360::fm238.subfm0
```

Figure 3.7 – Screen capture of the LatSeq fingerprints file.

thread access to such a buffer require an inter-thread synchronization. Thus, this would be inefficient.

The *data collector* is the low-priority LatSeq-specific thread. Assigning low-priority to this thread is conservative regarding CPU utilization and to minimize its impact on the BS. The data collector loop like measurement point, is optimized in terms of CPU instructions to meet high performance / low-impact requirements (See Appendix C.C.3). As illustrated in Figures 3.5 (also in Appendix C.C.1), this thread visits each ring buffer in a round-robin fashion. It gets one latseq element in a ring buffer and formats it to a string (one line) and goes to the next ring buffer. If the visited ring buffer is empty, then the collector goes directly to the next buffer. The data collector is the only thread that empties the ring buffers, and it runs asynchronously. The size of ring buffer is determined according to the thread that generates the most fingerprints. It is known that IO operations costs a lot for a program, in particular writing into a file.

The outcomes of the LatSeq measurement module is a flow of packet fingerprints written either on the standard output or a temporary file in a shared memory for further processing, illustrated with the screen capture in Figure 3.7. For future work we also propose to write the fingerprint flow to a network socket for a distant machine to process real-time the fingerprint flow.

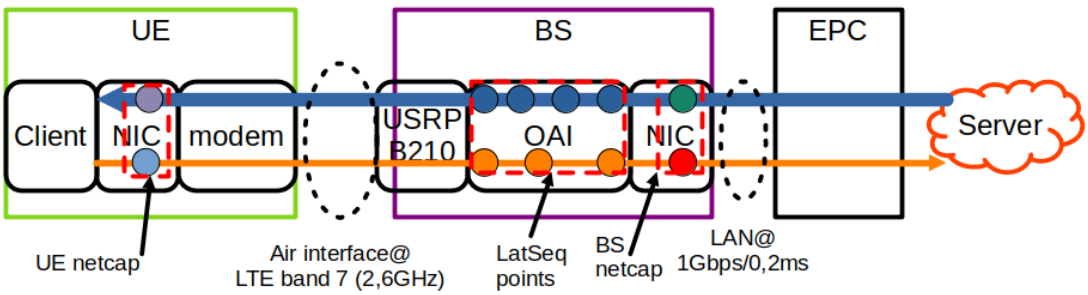



Figure 3.8 – LatSeq points in the RAN testbed.

### 3.3.2 Measurement points in the LTE stack

The implementation of LatSeq is done by adding measurement points `LATSEQ_P` in the OAI BS source code for the LTE stack. We instrumented the OAI code with 34 carefully chosen measurement points. Indeed, too many measurement points would be superfluous with not significant latency, as too few would make it impossible to rebuild journeys. Measurement points complete the latency measurement testbed presented in Fig. 2.4 with Fig. 3.8.

#### 3.3.2.1 Packet path in the LTE stack

LatSeq has been implemented in the LTE stack. There are 2 main packet paths, one from PDN to UE (i.e. in the downlink) and one from UE to PDN (i.e. in the uplink). In the downlink the input measurement point is set in the PDCP layer and the output in the PHY layer. In the uplink, the input is set in PHY and the output in the PDCP interface with GPRS Tunneling Protocol (GTP). The others in-stack LatSeq points are represented in the Appendix C.C.4. The number of measurement points depends at least on the number of layers because of local identifiers needed to link the input to the output.

The method we suggest putting useful measurement point for being able to rebuild the packet path is to define one entry function with the incoming data structure (for instance, the received bytes array from the antenna in the uplink) and one output function (for instance, the `sendmsg` function at the network interface). Following them, find functions that manipulate data structures and associated data identifiers. The necessary measurement points are those that link 2 different data structures containing user data or 2 different functions that manipulate a different set of data identifiers. When the minimal set of measurement points between entry and output is defined, more measurement points could be added to surround a specific network function, we want to know more about its

specific latency.

Let's take an example in the uplink:

```
1 1623775199.784367  U  phy.in.ant—phy.in.proc  len15360::fm281.subfm8
2 1623775199.784455  U  phy.in.proc—mac.harq.up  len66::ue0.cbseg0.fm281.
    subfm8
3 1623775199.784468  U  mac.harq.up—mac.demux  len30:rnti58106:ue0.lcid3.
    fm281.subfm8
4 1623775199.784468  U  mac.demux—rlc.rx.am  len30:rnti58106:drb1.lcid3.
    rsn138.rso0.rfi2.fm281
5 1623775199.784468  U  rlc.rx.am—rlc.reas.am  len30:rnti58106:drb1.rsn138.
    rso0.rfi2
6 1623775199.784469  U  rlc.reas.am—pdcp.rx  len86:rnti58106:drb1.rsn138.
    rso0.psn0.psn148
7 1623775199.784475  U  pdcp.rx—gtp.out  len86:rnti58106:drb1.psn148
8 1623775199.784502  U  gtp.out—ip.out  len84:rnti58106:drb1.psn148.teid1.
    gsn149.ipid0xe9e7
```

Listing 3.5 – Consecutive LatSeq fingerprint of a packet segment

With the following local data identifiers, we could not link directly to the same packet the first fingerprint with the eighth. But we could link the fingerprint 1. and 2. thanks to the common point `phy.in.proc` of the packet segment and thanks to local identifiers `fm` (frame number) and `subfm` (subframe number) which match between the two fingerprints. Sequentially, `lcid` (LCID), `rsn` (RLC sequence number), `rso` (RLC sequence offset), `psn` (PDCP sequence number) and `drb` (DRB id) are local identifiers to track packets between segments. Solely the fingerprint 7 is not necessary to link segments (as `psn` used to link RLC to IP layer through PDCP is already known at the frontier of RLC and PDCP layer with the segment `rlc.reas.am-pdcp.rx`) but all other points are necessary to track packet. The interest of `pdcp.rx-gtp.out` measurement point is to take a precise measurement of delay introduced by the `pdcp.rx` point. Note that at the output segment, we could have just indicated the `psn` value to link the last segment with the second-to-last segment. However, we recall that latseq is integrated on global package of end-to-end latency measurement where latency is measured with passive network captures. GTP and IP protocols are used to carry packet respectively in mobile core network and in packet data network. They define a tuple of values to identify packets and flows. In particular, the IP protocol defines the IP id (identification) field that is used to distinguish IP packets. The `ipid` flag is accessible before the GTP encapsulation (conversely after

the GTP decapsulation) at the measurement point since it has a fixed position in the IP header with the code:

```
uint16_t ipid = data_req_p->buffer[data_req_p->offset + 4] << 8 |
    data_req_p->buffer[data_req_p->offset + 5];
```

Listing 3.6 – IP ID catching at the GTP layer

Afterwards, we correlate the packet IP id of the packet fingerprint with the IP id from network captures, it provides an end-to-end continuity for packet tracking.

We distinguish *3 types of segments* in our implementation : Processing-related segments; Buffer-related segments; and Retransmission segments.

It is convenient to differentiate the type of delays. A latency due to a processing function could be reduced by optimizing the function or simplifying its task when a latency of a buffer is due to a service access (e.g. RLC buffer because of MAC scheduling to radio access). For that, we box the buffer with measurement points to isolate latency experienced by packets in buffers.

#### 3.3.2.2 Contextual measurement points

In an extension of LatSeq we added a new type of measurement points not related to packets but contextual, the type "**information**" `I`. Indeed, we found that radio context is really meaningful to analyse latency along with the internal latencies from fingerprints. For instance, the RLC buffer latency is related to the mac scheduling decisions and the retransmission latency is related to channel quality. The fingerprints generated by the implemented `I` measurement points are listed below:

```
1  1623775199.783456  I  mac.harq.up  nack0:ue0:harq1.fm281.subfm7
2  1623775199.783468  I  phy.srs  dcqi15:ru0.ue0:
3  1623775199.783468  I  phy.srs  ucqi165:ru0.ue0:
4  1623775199.783468  I  mac.ind  bsr8.len31:ue0:lcgid1
5  1623775199.784468  I  rlc.rxbuf.am  occ30:rnti58106:drb1
6  1623775199.784491  I  mac.sched.down  mcs28.tbs7.nrb3:rnti58106:lcid3
7  1623775199.784805  I  rlc.txbuf.am  occ86:rnti58106:drb1
8  1623775199.796435  I  mac.ind  sr1:ue0:fm283.subfm0
9  1623775199.796445  I  mac.sched.up  mcs10.tbs63.nrb3:ue0:fm283.subfm4
10 1623775199.804465  I  mac.ind  phr40:ue0:
```

Listing 3.7 – Radio context capture with LatSeq "I" fingerprints

As for packet-related fingerprints, the "I" fingerprints is composed of a timestamp, a location in the stack and data properties. The information fingerprints had revealed very useful to enlighten the latency last segment not covered by LatSeq i.e. the UE air interface modem.

The work to implement measurement points should not be played down. It is a work of an expert of the radio stack that knows packet path, data structures, functions and standards. But the implementation of itself is easy, with only a macro instruction LATSEQ_P with the suitable arguments to put inside the source code.

### 3.3.3 Internal latency analysis

The output of the LatSeq measurement is the packet fingerprint file, including a flow of fingerprints related to different UEs, layers, packets, etc. The rebuilding phase (Fig. 3.5-(C)) where fingerprints are put together to rebuild packet journey is necessary to compute internal latency. Latency data are then filtered and plotted to provide a comprehensive analysis of the internal latency.

#### 3.3.3.1 Packet journey rebuilding algorithm

The main task of the analysis module is to *rebuild the journeys* of the different packets, associating each journey with a graph of causally linked fingerprints. The configuration of the rebuilder defines the input points (e.g. ip.in for the downlink, phy.ant.in for the uplink) and the output points (e.g. phy.out.ant for the downlink and ip.out for the uplink). The rebuilding algorithm tries to find a path between the input and the output points constituted of fingerprints (edge in the graph vocabulary) that match with data identifiers and time causality.

The algorithm is as presented in Algorithm 1 [1]:

When a fingerprint that corresponds to an entry segment is found, then a new journey is created (RECURSIVE_BUILD). For a first segment as A-B, the next segment should be B-X. The list of fingerprints is unfolded when fingerprint with a segment match with an unfinished journey, the local identifier are compared. The global identifier quickly filter fingerprints that not belong to journey. All the local identifiers known for the journey should match with the local identifiers present in the fingerprints in a logical AND

1. Algorithm python code in Github: https://github.com/Orange-OpenSource/LatSeq/blob/master/tools/latseq_logs.py

**Algorithm 1** Rebuild packet journey

```
Require: I                                                  ▷ List of input points
Require: O                                                  ▷ List of output points
  procedure REBUILD_JOURNEYS(fingerprints)
    F ← fingerprints                                        ▷ List of fingerprints ordered by timestamp
    J ← dict()                                              ▷ Dictionary of output journeys
    while F.head is not empty do                            ▷ For all fingerprints in the file
      while F.head not in I do                              ▷ Find a new possible journey start
        F.head ← F.head.next
      end while
      f ← F.head                                            ▷ A new input fingerprint
      G ⊂ F                                                 ▷ Subset of F where the journey should be completed
      Initialize a new journey empty j
      J ← J ∪ RECURSIVE_BUILD(G, f, j)
      F.head ← F.head.next                                  ▷ Proceed to the next fingerprint
    end while
    return J
  end procedure
  procedure RECURSIVE_BUILD(G, f, j)
    while G.head is not empty do
      if G.head.src = f.dst then                            ▷ Candidate for next journey
        m ← MATCH_M(G.head.gid, f.gid, G.head.lid, f.lid)   ▷ gid: set of global identifiers, lid: set of local identifiers
        if m is empty then                                  ▷ data identifiers do not match
          continue
        end if
        S ⊂ G                                               ▷ Subset of G where to find possible segmentation
        Find possible segmentation f′ in S
        j ← j ∪ G.head                                      ▷ Add the fingerprint to the journey
                                                            ▷ We cannot remove G.head because of concatenation case
        if G.head in O then
          J′ ← j
          for all h in f′ do                                ▷ Rebuild all segmentation journeys
            J′ ← J′ ∪ RECURSIVE_BUILD(S, f′, j)
          end for
          return J′
        end if
        G.head ← G.head.next
      end if
    end while
    return empty                                            ▷ Journey incomplete
  end procedure
  procedure MATCH_M(g.gid, g.lid, f.gid, f.lid)             ▷ Classify the fingerprint in the current journey
    if g.gid ∪ f.gid then
      return empty
    end if
    locals ← (g.lid ∪ f.lid)                                ▷ Set of local identifier in common between f and g
    for all l in locals do
      if g.lid[l] ≠ f.lid[l] then                           ▷ On local identifier value is different, not in the journey
        return empty
      end if
    end for
    return locals
  end procedure
```

style. Any local identifier mismatch means the current fingerprints does not belong to the journey (MATCH_M). A journey is considered completed as soon as the output point is reached.


```
{
    "uid": "30",
    "dir": 0,
    "file": "/tmp/main_ocp.01012022.lseq",
    "ts_in": 1604086525.332053,
    "ts_out": 1604086525.332791,
    "path": 0,
    "completed": true,
    "properties": {"len": "62"},
    "glob": {"rnti": "53421"},
    "set_ids": {"uid": "30", "teid": "4", "gsn": "253", "drb": "1", "psn": "88", "sdu": "88", "rsn": "72", "rso": "0", "lcid": "3", "reqfm": "686", "txreq": "0", "harq": "3", "sfn": "11701", "fm": "731", "subfm": "5"},
    "set": [[33, 1604086525.332053, "ip.in—gtp.in"], [34, 1604086525.332054, "gtp.in—pdcp.in.gtp"], [35, 1604086525.332054, "pdcp.ip—pdcp.tx"], [36, 1604086525.332055, "pdcp.tx—rlc.tx.am"], [40, 1604086525.332481, "rlc.tx.am—rlc.seg.am"], [41, 1604086525.332482, "rlc.seg.am—mac.mux"], [42, 1604086525.332483, "mac.mux—mac.txreq"], [43, 1604086525.332484, "mac.txreq—mac.harq"], [44, 1604086525.332509, "mac.harq—phy.out.proc"], [45, 1604086525.332791, "phy.out.proc—phy.out.ant"]]
}
```


Listing 3.8 – Rebuilding output of a journey in a JSON format

The outcome is a line per journey in a JavaScript Object Notation (JSON) format as presented in Listing 3.8 that contains among other things the related local identifiers and timestamp associated to segments. For this downlink journey, the internal latency is 0.738 ms and the 2 segments that contribute the most to the latency is the RLC transmission buffer and the PHY encoding process.

The algorithm 1 complexity is $O(n)$, where $n$ is the number of fingerprints when fingerprints are ordered and all packet journeys belong to the "linear" case. The complexity of the algorithm grows with the number of segmentations and concatenations (Fig. 3.3) that are present in the cellular networks. Indeed, contrary to the linear case where we can definitively complete a journey when the output point is reached, segmentation, concatenation or retransmissions generates recursive journey rebuilding. The number of journeys to consider for a new fingerprint grows exponentially with the size of the fingerprint file. The classification of a fingerprint to a journey is quick at the start of the fingerprint flow but very slow at the end when we have to loop over thousands of journeys to consider for concatenation, segmentation and retransmission. We solve the problem with configurations for the algorithm that reduce the research space. Three configuration values limit the search space in time, respectively, the concatenation (100 ms), the segmentation (100 ms) and the retransmission (32 ms). The values are chosen to limit the error of rebuilding while limiting as low as possible the search space. For a new fingerprint, we

update, with the timestamp the set of journeys that are open for a new segmentation, a concatenation or a retransmission. Thus, the number of journeys to look at for a given journey is contained during the processing and does not explode. Note that the algorithm allows to rebuild "on-the-flow" by feeding the rebuilding algorithm with the measurement part output.

#### 3.3.3.2 Filtering results

The LatSeq measurement part generates fingerprints for all data units forwarded by the BS. Should we want to restrain observation to a subset, for example, to a given family of journeys (e.g. a flow), then a **filtering** module is required. Implementing this filter on the measurement side is not possible, as the journey to which each fingerprint belongs is calculated later, in the analysis part. The filter module should necessarily be implemented after the rebuilder. With adequate filters in the analysis module, it is possible to investigate the behaviour of a protocol layer, a specific UE or a mechanism (*e.g. segmentation*). A wide range of filters can be defined. We give a few examples below:

— By selecting journeys with the same RNTI, the behavior of a specific UE can be studied.
— By additionally filtering on a local identifier (e.g. an IP packet number), the packet's entire life can be explained.
— The internal latency for a given bearer can be measured with a filter on a DRB value.
— The behaviour of any layer can also be analyzed with a filter on a `src` or `dest` value.

#### 3.3.3.3 Visualization and statistics generation

We developed a set of scripts to compute *statistics* and tools for *visualizations*. In particular, a tool to fill the matrix with internal latencies in Fig. 3.4. The output is a CSV file containing internal latency per layer, per journey, per UE. Standard visualization tools such as *gnuplot*[2] could then be used to plot internal latency for instance in Figures 3.9. The cumulative volume of internal latency for journey and the contribution of the RLC layer in the downlink and in the uplink.

2. `http://www.gnuplot.info`

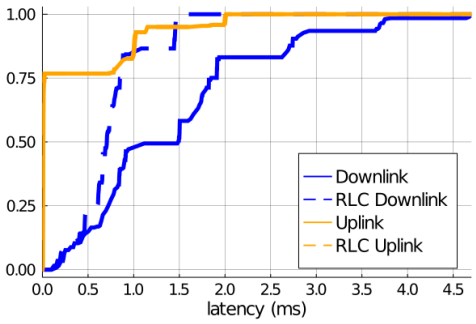



Figure 3.9 – ECDF of total internal latency and latency in the RLC layer for Downlink and Uplink.

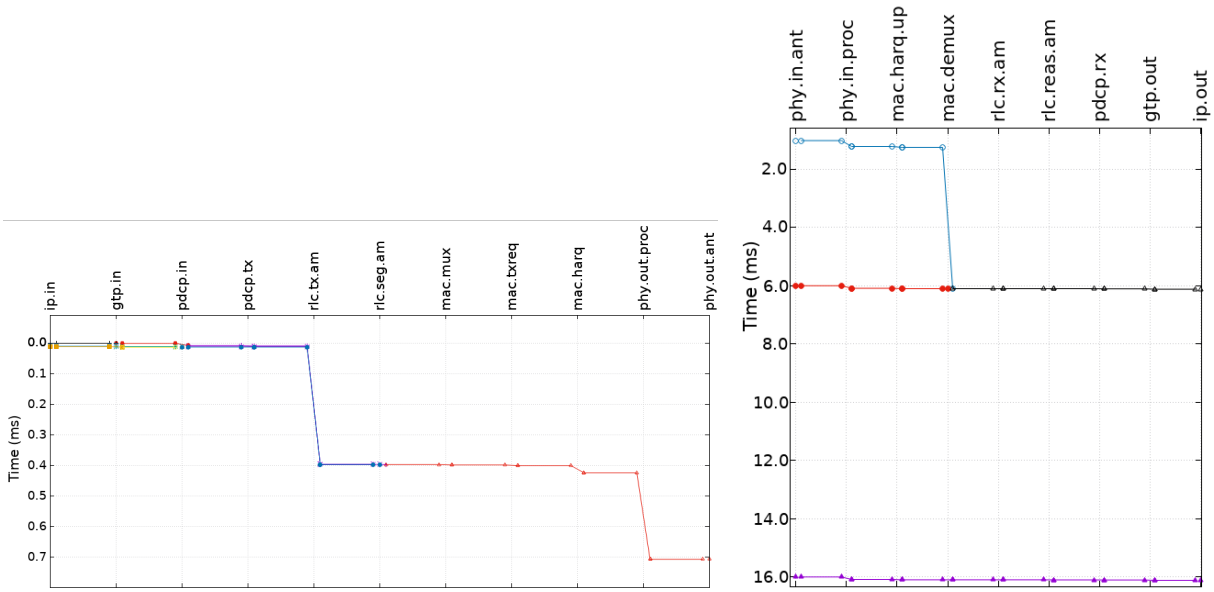
An important contribution of our approach is the **waterfall** plot as a powerful representation (see Figures 3.10). This type of plot enlightens latency causality in a packet journey in a time-sequence plot. A small part of uplink transmissions is isolated showing the packet journey of 2 IP packets but 3 radio transmissions. The first IP packet has been transmitted over 2 radio frame and reassemble at the RLC layer. For the two IP packets, the decoding delay at `phy.in.proc` segment constitutes a significant part of the total uplink latency. This visualisation is a valuable tool not to study the latency in the all but for a deep understanding of radio transmission phenomena on a small set of selected journeys, for instance, to debug high tail latency.

In this work, we focus on latency but LatSeq opens the door of a wide variety of analyses. For example, thanks to the `len` field, the instantaneous throughput can be computed, possibly at each layer or for each user. Extra information can be added in the `properties` field of the fingerprint to perform analyses based on other criteria. Also, in an extension of the tool, we added the `"I"` fingerprints which gather some information of the radio context. Especially, radio latency is correlated to buffer occupancy, scheduling decision, and retransmissions. We can put side by side the waterfall of the packet transmission with the downlink and uplink information (see Appendix C.C.5).

In the downlink, we plot the RLC transmission buffer occupancy, the reported channel quality, the MCS corresponding to scheduling decision and HARQ retransmissions.

In the uplink, we plot also the RLC reception buffer occupancy, the computed channel quality (Channel Quality Indication (CQI)), the BSR and SR that are related to uplink scheduling decisions and HARQ retransmissions.

These representations are not new and are somewhat equivalent to some debugging tool of Amarisoft but interesting when putting in regard with latency waterfall.




(a) One packet in the downlink.

(b) Two packets in the uplink.

Figure 3.10 – Waterfall representation of packet journeys.

| **Experiment** in low-load conditions | Unit | **Baseline** | | **LatSeq** | |
|---|---|---|---|---|---|
| | | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| Fingerprint generation time with 1 argument | ns | 0 | | 10.98 | 0.2 |
| Data collector memory usage | kB | 0 | | 450 | |
| Linux load average | % | 0.79 | 0.03 | 0.81 | 0.03 |
| Linux load average in loaded condition | % | 1.18 | 0.065 | 1.25 | 0.075 |

Table 3.2 – LatSeq impact on BS performance results. $\mu$ *: mean,* $\sigma$ *: standard deviation*

## 3.4 Evaluation

A major point for the relevancy the proposed method of measurement is its low impact on the BS performances even while recording a huge amount of packet fingerprints. The throughput could attain Gbps or more than 100.000 data packets per second in a 5G BS 100.000 packets per second with 10 measurement points on the path means it could be generated 1.000.000 fingerprints per second.

The rebuilding phase is not time-constrainted contrary to the measurement in the BS. We evaluate the delay added by the measurement points on the packet path and the cost on BS performances of the fingerprint generation.

Performance results are summarized in Table 3.2. The fingerprint generation experiment (subsection 3.4.1) evaluates the measurement part (Fig. 3.5-(A)), with the time to generate a fingerprint (in the table with one data identifier i.e. variadic macro-instruction argument). The data collector part (Fig. 3.5-(B)) is evaluated both in terms of memory usage (subsection 3.4.2) and CPU usage (subsection 3.4.3).

### 3.4.1 Unitary measurement point generation time

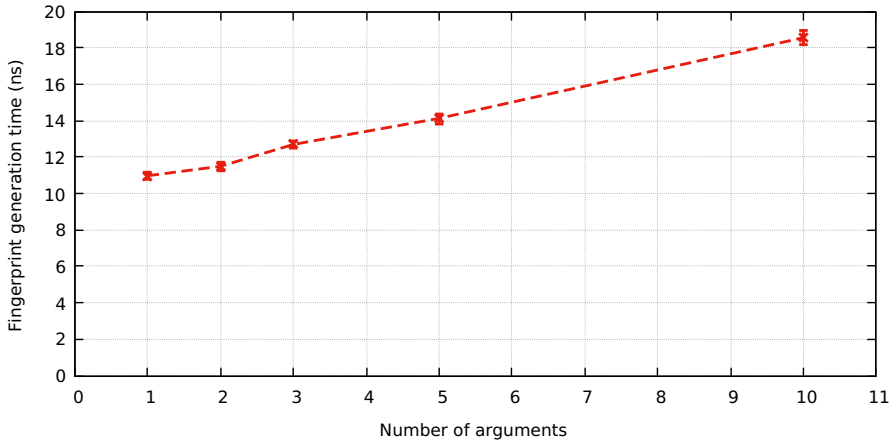



Figure 3.11 – Time to generate a fingerprint according to the number of arguments.

An analysis of the assembly code generated shows we add 37 machine code instructions per LatSeq measurement points (Appendix C.C.2). On a unit test setup, we measured the execution time of the LatSeq macro on 1 million runs from 1 to 10 arguments (*e.g. data identifiers*). The execution time according to the number of arguments is plotted in Figure 3.11. The average time goes from 10.98 ns for 1 argument to 18.56 ns for 10. On a CPU with a frequency of 3.0 GHz, 14 ns (5 arguments) is equal to 42 CPU cycles [3] that is not so far from the 37 found in Appendix C.C.2. The increase of the execution time is linear to the number of arguments. In fact, the number of assembly instructions associated to the copy of one identifier to ring buffer is fixed, and add a fixed amount of execution time (Listing 3.4). In conclusion, for a given packet measured at 10 different points along the path, measurement points incurs of 200 ns (for 10 data identifiers), 5 times lower than our objective of a microsecond resolution.

3. `x86` instruction table : `https://www.agner.org/optimize/instruction_tables.pdf`

### 3.4.2 Memory usage

Concerning the memory, it is allocated a ring buffer of typically 256 latseq measurement elements per instrumented thread. An element is a data-structure of 180 bytes, thus the ring buffer size in memory is of 46 kB. The number of instrumented threads depends on OAI source code structure, but it is linked to the number of MAC/PHY encoding/decoding process and the number of data radio bearers. OAI BS instantiates 8 threads for MAC/PHY (linked to the number of HARQ processes), and one thread per data radio bearer. The number of ring buffers is then equal to 8 + the number of DRB. By default, one default bearer is instantiated at the connection establishment of a UE. Then the size in memory of LatSeq measurement is directly linked to the number of connected UE. For one UE, the size in memory of LatSeq is then about 450 kB which is a limited impact on memory requirements.

### 3.4.3 In-running global performance

To evaluate LatSeq contribution to the global performance, we measure its CPU utilization (with `perf` kernel tool) when fingerprints are generated at full speed (i.e. max disk writing speed of 264 MBps in our setup). The screen capture in Appendix C.C.6 shows a part of 0.22% of LatSeq on the total cpu usage by the BS program. Thus, the collector thread is not CPU-bound (usage $< 100\%$). Moreover, the BS's linux load average (that comprises the CPU and IO operation) is not so impacted by the presence of LatSeq in both baseline and loaded condition, respectively +0.02 (with 0.03 of incertainty) and +0.07 (with 0.075 incertainty). LatSeq has then a visible impact on machine performances but limited and principally due to the data collector's capacity bottleneck that is on the writing-to-file side, not on the fingerprint-formatting one. We can then roughly estimate LatSeq's capacity as the maximum writing speed in ring buffers, that is about the full disk writing speed. With disk writing speed of 264 MBps, for a fingerprint size of 100 bytes, it corresponds to about 2.64 M fingerprints per second.

Over the assumption that each packet forwarded by the BS generates 10 fingerprints, then the maximum throughput we can measure accurately is $64 \times 2.64.10^6/10 = 16.9$ MBps (in both side) in the (worst) case of 64-bytes packets. Moreover, as a running BS has no need for disk access, LatSeq does not threaten the system's performance even at full disk throughput. A manner to accelerate LatSeq is to write fingerprint on the high-speed RAM with shared memory. The fingerprints are then processed and filtered

(e.g. per UE) from the shared memory by rebuilding and filtering scripts and the results is finally written on the disk. We also propose as further work to write fingerprints on a network interface dedicating to BS monitoring with a server which receives fingerprints.

## 3.5 Related work

The idea of measuring latency inside the RAN partly was inspired by [188] where they measure in-device latency. The measurement method and technical solutions was inspired by several works in [189–192]. These solutions described mainly high performances logging systems not specific for cellular networks.

A number of tools are provided with the OpenAirInterface project. Two of them are to be cited for the packet measurement and network monitoring, **T_tracer**[4] and **Wireshark captures**. T_tracer is a framework to debug and monitor softmodem. With a set of measurement points inside the BS, T_Tracer pushes data to monitoring tools. The monitoring tool called *enb*[5] provided traces of data unit at PHY, MAC, RLC and PDCP layers. We did not use the measurement part of T_tracer because it has been designed for debugging and monitoring and not for the performances.

Another module of T_Tracer is the possibility to put passive capture point for *wireshark*[6] directly in the LTE stack. The measurement point copies the content of a packet to a local port where wireshark, a graphical network capture software could listen and capture LTE packets. In our setup it would have more passive measurement points but inside the RAN segment that is not possible otherwise. A clear limitation is that to have the same complete view of internal like with LatSeq we should multiply the number of passive network capture points, but these captures have a high cost. When a capture is performed, even if we do not count the measurement point cost in itself, the whole packet is captured with the payload. Even with a smart truncation of the packet to keep only the header associated to the layer, the cost in terms of CPU instruction is always higher than what we propose with LatSeq fingerprints capture. In addition to packet header as local identifier, LatSeq indicates the packet segment of the fingerprints. The rebuilding could be made without any knowledge of the underlying packet path architecture when with network capture, the packet path should be indicating as a configuration. The T tracer solution is to be quoted because it is a powerful tool for BS debugging but does not meet the logging performances required for realistic experimentation not in a debugging mode.

4. `https://gitlab.eurecom.fr/oai/openairinterface5g/-/wikis/T`
5. `https://gitlab.eurecom.fr/oai/openairinterface5g/-/wikis/T/enb`
6. `https://gitlab.eurecom.fr/oai/openairinterface5g/-/wikis/T/wireshark`

## 3.6 Showcasing LatSeq: TCP ACK packet bursting in uplink

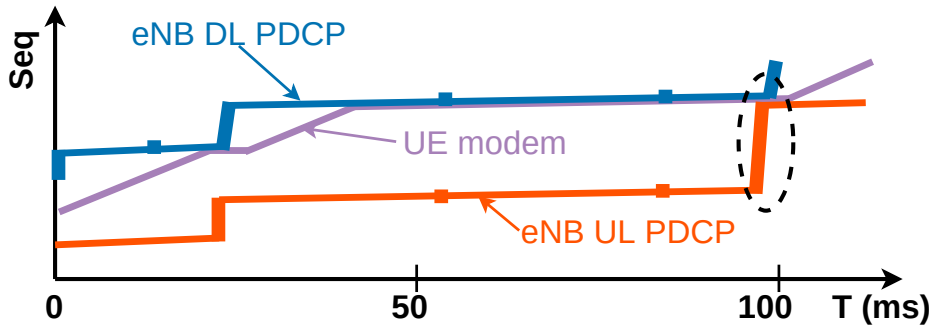



Figure 3.12 – Traffic profile of the LatSeq showcasing experiment. *Blue: At the entry of BS in DL; Purple: At the UE; Orange: At the output of the BS in UL. The burst of transmission to debug is encircled*

In this section, we showcase the capabilities of LatSeq to effectively exhibit internal latencies inside BS in coordination with the passive network capture. We observed in deployed cellular networks an effect of bursty transmissions of packets, particularly marked in the uplink. We believe it is relevant to explore these phenomena on our testbed (presented in Chapter 2) with LatSeq.

On the presented testbed, four passive capture points are put on the packet path as illustrated in Fig. 2.4. A downlink TCP traffic is generated from server to terminal which generates in the uplink a flow of acknowledgments. Each acknowledgment is typically a 100-bytes IP packet. The TCP time-sequence from network captures is plotted in Fig. 3.12.

The data packets and their corresponding acknowledgments are represented with the cross. Three problems had to be highlighted here:

1. The high latency in the uplink compared to the downlink
2. The uplink delay jitter from one packet to another
3. The bursty transmission of packets in the uplink when in the downlink the packet are regularly transmitted

The answer to this question cannot be resolved nor with the network captures only neither with latency statistics from the BS. With LatSeq measurements the situation

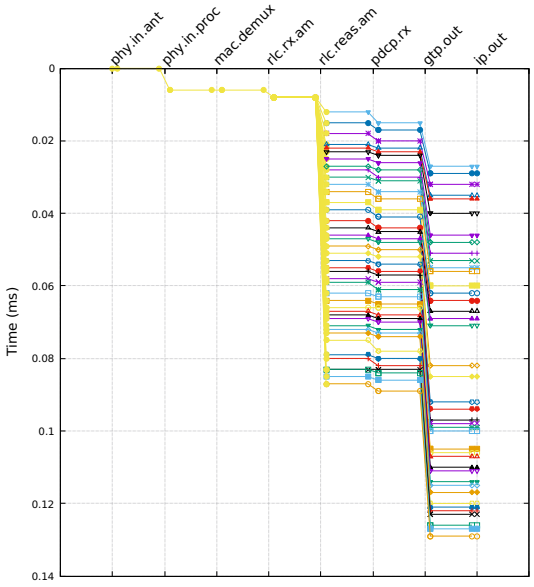



Figure 3.13 – Waterfall of a burst of transmission in the uplink.

become clearer compared to the black box represented by the large white empty space on the time-sequence plot.

When we zoom on the waterfall of the transmission burst, we found that all acknowledgment packets are transmitted in the same transport block (in yellow on Fig.3.13). Then all packet issues to the same uplink grant. The reassembling of the IP packets are made at the RLC layer, confirming that all packets are packed on the UE side in the transport block. The packet journeys do not show HARQ or RLC ARQ retransmissions. We also eliminate the possibility of a RLC in-sequence delivery because there are no concatenation with an earlier transport block.

The radio context confirms that there are no disturbances in the air interface that could incur decoding error and retransmission. However, the explicit scheduling request (SR/BSR) sent by the UE are correlated with the transmission by the BS of an Uplink Grant or grant of uplink capacity (UG). The size of the grants transmitted after a BSR reception allow the transmission of several packets especially since the acknowledgment packets are small (around 100 bytes). In this example, the uplink latency should be correlated with the radio resource allocation with the transmission of an UG.

As partial conclusion, LatSeq demonstrates on this short example how it could help the analysis of latency in the RAN. In addition to radio context information, it is a very powerful tool for debugging but also to get relevant latency statistics per layer.

## 3.7 Contribution to open source community

This work would not have been possible without an open source RAN or an open source OS or even without the open source community around OpenAirInterface. As a consequence, and in the pure philosophy of open source, LatSeq is made available to everyone with a permissive license (BSD 3-clause) for any other project. The implementation is made as independent as possible from the OAI project and the sources and examples are available on Orange open-source github group [7]. We had already some contacts with external people about the tool and how to use it with great interest. The wide range applicability of the solution thanks to its general concept and the different communications made in intern at Orange and in external at conferences should contribute to the distribution of the tool.

In a second time, we pushed to the official OAI repo the LatSeq branch [8] of the implementation of measurement points to be ease the usage of the tool by the OAI community. We started the process after a presentation at the OAI summer workshop that showed great interest from the OAI community, especially for 5G Ultra-Reliable Low-Latency Communications (URLLC) developments. We collaborate with Eurecom to implement LatSeq to the OAI 5G stack [9] since we implement it for LTE only in a first time. The hope to have the tool to survive after this thesis is a main contribution of this work for the Software-Defined RAN (SD-RAN) open source community.

7. `https://github.com/Orange-OpenSource/LatSeq`
8. `https://gitlab.eurecom.fr/oai/openairinterface5g/-/merge_requests/1432/`
9. `https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/latseq_5GSA`

## 3.8 Conclusion

In this chapter we present a novel approach to measure latencies inside a software BS. Based on definitions of the *packet fingerprint* as the measurement input, the *packet journey* for the path of a packet and the matrix of *internal latency* we implemented tool called LatSeq.

This tool is composed of a low-impact measurement extension for the OpenAirInterface BS and a module to process and visualize the journeys of packets and various statistics. This paves the way to a very wide range of fine-grained and comprehensive internal latency analyses as illustrated with the matrix of delay (Fig. 3.4). LatSeq provides for an understanding of interactions between the different layers in terms of delays especially with the usage of the waterfall representation of delays (Fig. 3.10a). We showcase this ability on a real-world experiment. This makes LatSeq a valuable help to optimise LTE and 5G radio interfaces.

It should be noted that LatSeq is sufficiently independent of the OAI platform's code and structure to be generalized to a wide variety of open-source software, *e.g. srsLTE*. The code of the tool and its implementation on OAI has been released on open-source. This is an important aspect for the dissemination of this work and a contribution to the OAI community.

Chapter 4

# The Causes of Latency in the Radio Access Networks

**Abstract** : *Thanks to the experimental platform and comprehensive latency measurement tools that we have put in place, we experimentally evaluate the sources of latency in the Radio Access Network (RAN). We first present a set of experimental evaluations to explore latency in the OpenAirInterface (OAI) lab testbed for an error-free stable radio-channel condition. We found that the uplink channel is responsible for important latency and jitter. We investigate in the last section this issue in depth and demonstrate the role of the uplink radio resource allocation scheme based on SR, BSR and UG. We put this observation on the experimental testbed with a trial in an operational cellular network and literature.*

## Contents

**4.1 RAN Latency Characterization in the Lab Testbed . . . . . 95**
4.1.1 Baseline RAN RTT with Ping . . . . . 96
4.1.2 Connection establishment, RTT and HTTP/S transfer . . . . . 101
4.1.3 Exhibiting bufferbloat . . . . . 104
**4.2 Uplink Segment as a Source of Latency and Jitter . . . . . 108**
4.2.1 Uplink transfer experiment . . . . . 108
4.2.2 Uplink latency and jitter in a commercial RAN . . . . . 110
4.2.3 Analysis of the uplink radio resource allocation scheme . . . . . 114
4.2.4 Influence of RAN configurations . . . . . 118
4.2.5 On the buffer knowledge . . . . . 123
4.2.6 Related observations . . . . . 124
**4.3 Conclusion . . . . . 128**

**Publication**

— *On Radio Access Network Uplink Latency and Jitter : Measurements and Analysis*. **Flavien Ronteix–Jacquet**. IEEE ITC-33, September 2021, Valence / Virtual, France. Link to paper : `https://hal.archives-ouvertes.fr/hal-03420681`

| Experiment | Section | Protocol | Latency characteristics |
|---|---|---|---|
| **Baseline RTT** | §4.1.1 | ICMP | RAN RTT<br>UL/DL One-Way Delay (OWD) |
| **Packet size and RTT** | §4.1.1 | ICMP | RAN RTT<br>Segmentation and Concatenation |
| **RTT in a loaded bearer** | §4.1.1 | ICMP / UDP | RAN RTT<br>Network load |
| **Short HTTP transfer** | §4.1.2.1 | TCP | Transfer duration,<br>RRC connection delay |
| **Long HTTP transfer** | §4.1.2.2 | TCP | Transfer duration<br>RTT variation |
| **RRUL** | §4.1.3 | ICMP / TCPs | Uplink bufferbloat<br>Asymmetry DL/UL |
| **Uplink long transfer** | §4.2.1 | UDP | OWD uplink<br>Uplink latency jitter |

Table 4.1 – Summarized latency experiments.

## 4.1 RAN Latency Characterization in the Lab Testbed

It could be found in the literature many work about latency measurements in real-world mobile networks [16–18, 20, 22, 193–196][93]-3.1. Also, the latency is frequently evaluated as a performance indicator in simulation studies when it is not the center of the focus. Our lab testbed allows going deeper in the analysis of per-packet latency on a selected set of internet traffic in a Best-Effort configuration. This work is a prior to the thesis direction followed thereafter.

This section is a baseline to understand latency related to OAI RAN segment. Table 4.1 summarizes the experimental trials that was performed. Firstly, an evaluation of the RAN RTT with ICMP pings was performed for a latency baseline. Internet traffic is in majority composed of HyperText Transfer Protocol (HTTP)/TCP/IP flows. Experiments with short and longer HTTP transfer are used to evaluate the transfer pattern and duration of such internet flows. Finally, UDP long transmission and *Flent* experiments exhibit the well-known bufferbloat issue we encounter in the RAN.

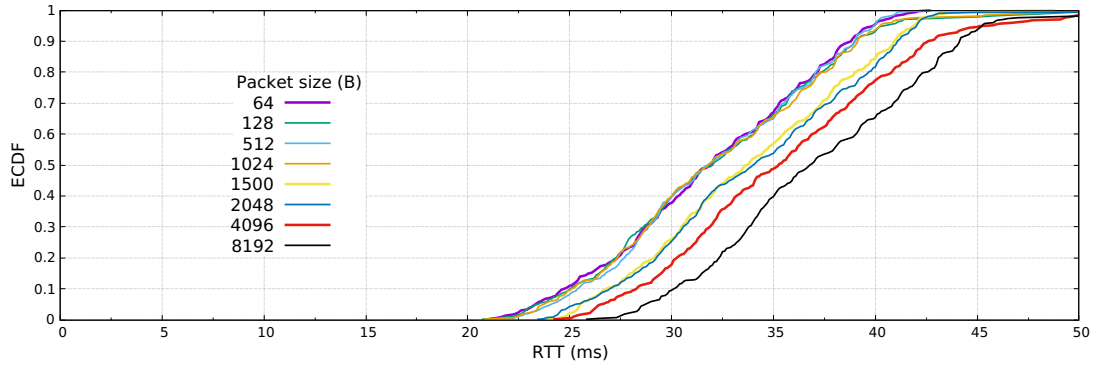



Figure 4.1 – Ping RTT for different packet sizes in an empty RAN.

### 4.1.1 Baseline RAN RTT with Ping

A very popular tool among network testers is the ping command [1] This command uses ICMP protocol to send a ping request to a target, the target respond with a ping reply that gives the network RTT. The particularity of ping is that the request is directly handled by the OS kernel, which avoids as much as possible application delays (see Fig. 1.8). The results of this section are used as baseline latency of the RAN (i.e. RTT, downlink/uplink OWD) for this thesis according to the RAN configuration presented in Table 2.2.

Different parameters are affecting packet latency :

— The *positioning* of the user in the cell modifies the received signal strength. A low or varying signal strength increases the probability of radio loss (with HARQ error recovery latency) and diminishes the MCS and the achievable throughput and thus, increases the packet transmission time.

— The *content positioning* with respect to mobile user affects the packet path length (i.e. propagation delay).

— The *scheduling strategy* may decide to prioritize latency over throughput or prioritize inter-user fairness over latency with a big impact on the resulting latency.

— The *packet size* is also determinant in the latency because a larger packet will take several TTIs to be delivered when a small packets could be successfully delivered with few Physical Resource Blocks (PRBs).

— The packet *datarate* is the source of queueing latency when it is higher than the radio channel capacity. Especially, queueing latency shows a peak when a burst of packets arrives with a high instantaneous datarate.

1. "How to measure network latency" : `https://kadiska.com/measure-network-latency/`

| Step | Value |
|---|---|
| BS proc. | 1 ms |
| TTI align. | $t_{\mathrm{FA}}$ ($< 1$ ms) |
| Data TTI | 1 ms |
| UE proc. | 1.5 ms |
| HARQ retx | $t_{\mathrm{HARQ}} \times P_{\mathrm{BLER}}$ |
| Total downlink (ms) | $3.5 + t_{\mathrm{FA}} + t_{\mathrm{HARQ}} \times P_{\mathrm{BLER}}$ |

(a) In the Downlink

| Step | Value |
|---|---|
| SR align. | $t_{\mathrm{SR}}$ ($\approx 10$ ms) |
| SR tx | 1 ms |
| BS sched. | 3 ms |
| UG tx | 1 ms |
| UE proc. | 3 ms |
| Data TTI | 1 ms |
| BS proc. | 1 ms |
| HARQ retx | $t_{\mathrm{HARQ}} \times P_{\mathrm{BLER}}$ |
| Total uplink (ms) | $10 + t_{\mathrm{SR}} + t_{\mathrm{HARQ}} \times P_{\mathrm{BLER}}$ |

(b) In the Uplink

Table 4.2 – User-plane One-Way Delay (OWD) in LTE RAN.

— The *Network load* causes the reduction of radio resources allocated per user and inter-cell interference this may cause a reduced Signal-to-Noise Ratio (SNR) and hence a lower MCS. Network load impacts the achievable throughput.

**Baseline RTT** In Figure 4.1 the ECDF of ping RTT for different packet sizes (7.000 points) is plotted for the testbed without any other load. A packet of 64 bytes corresponds to the default ping packet size. Packets are sent with an inter-departure time of 500 ms, resetting the network between each packet sending. The RTT ping median is 30 ms, the average is 32 ms, the variance is 5 ms, the minimum RTT is of 20 ms and 90% of packets experience a RTT of less than 40 ms.

The minimum value of 20 ms can be confirmed by an analysis of the different operations in the downlink and the uplink. LTE user-plane OWDs in the downlink and the uplink are presented in Tables 4.2, where $t_{\mathrm{FA}}$ is the time for the frame alignment before transmission, $t_{\mathrm{SR}}$ is the time before the transmission of a scheduling request, $t_{\mathrm{HARQ}}$ the time to recover from a radio loss with HARQ (e.g. 8 ms in LTE) and $P_{\mathrm{BLER}}$ the probability of radio loss. The minimum OWD is 3.5 ms and 10 ms for downlink and uplink, respectively. This gives a minimum RTT of 13.5 ms. The remaining delay is split between 0.5 ms for the backhaul/core/server latency and 6 ms inside the UE modem. HARQ process adds at most $8 \times 3 = 24$ ms before failing to recover radio loss. $P_{\mathrm{BLER}}$ target for Link Adaptation (LA) is usually 10%. Thus, the expected value of OWD due to HARQ is 0.888 ms ($<<$ 24 ms). HARQ introduces a jitter that was well studied in [197]. The

remaining loss ($< 1\%$ after HARQ process) are recovered by the RLC AM entity at the `t-reordering` timer expiration.

**RTT according to the packet size** In [198], Rekoputra evaluates the latency in an OAI RAN for different configurations. He varied the packet sizes and the ping packet inter-departure in a low-loaded RAN. He founds a high jitter and a difference of delay for different packet sizes. Furthermore, he compared the "baseline" ping with a Cloud-RAN (C-RAN) version of OAI and concluded that C-RAN leads to a better latency performances with lower latency. We do not have the C-RAN version he ran. ECDFs of RAN RTT presented in Figure 4.1 according to the packet size (we vary the packet size in uplink from 64 to 8192 bytes) shows different results concerning the increase of the RTT in respect with the increase of the packet size.

From 64 to 8192 Bytes, minimum, median and $90^{th}$ percentile RAN RTT increases of about 5 ms.

For the $99^{th}$ percentile packet RTT (higher 10 packets latency per size) the latency is not an increasing function of the size. Like every rare event, these high values are difficult to analyze, but, thanks to network captures and latseq traces, we know that is a bad scheduling in UE kernel or modem, decoding errors or the need of a connection re-establishment (with a random access procedure).

Also, for a packet size below the Maximum Transmission Unit (MTU) size of 1.472 bytes [2] where the RTT experienced does not seem to be impacted by the size of the packet. For packet sizes over the MTU, the uplink IP packet is segmented. Bigger than 1500-bytes packets are segmented at the UE Network Interface Card (NIC) into multiple IP packets. Packet bigger than 1472 bytes at the mobile networks ingress interface are segmented because of the GTP encapsulation into multiple GTP/IP packets. The maximum uplink Transport Block Size (TBS) for a MCS = 20 is 2384 bytes. Thus, a packet with a size inferior to 2384 bytes should be transmitted in one TB. For bigger packets that could not fit in a TBS, packets are segmented into several SDUs carried by several TBs at the UE RLC layer. Successive transport blocks are transmitted with an interval of 1 ms corresponding to the TTI duration.

We illustrate the operation of segmentation/transmission/reassembling with the example of a 4500-byte IP packet on the uplink. The packet is segmented in 3 MTU size IP

2. There are 28 bytes of header overhead to remove from the MTU because of GTP/IP encapsulation in the RAN

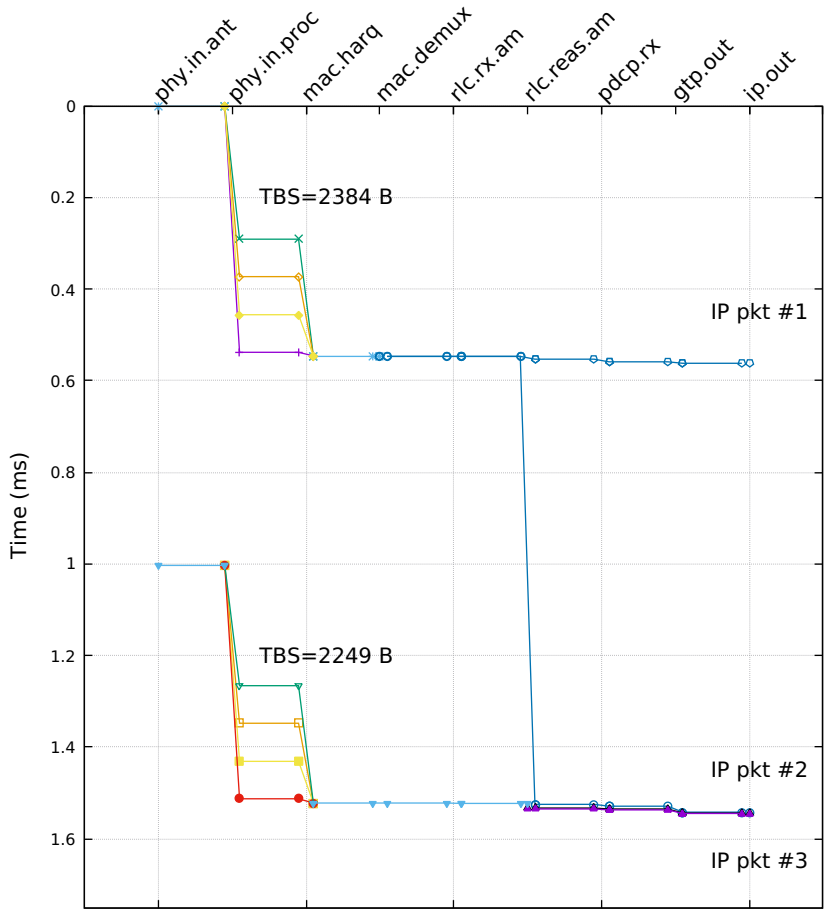



Figure 4.2 – Packet journey on the uplink for a 4.500-bytes IP packet.

packets by the UE NIC and transmitted within at least 2 TB through the air interface.

The packet journey on the uplink inside the base station is plotted in Figure 4.2. We observe that 2 successive radio subframes (i.e. 1 ms interval) carry 2 transport blocks of size 2384 and 2249 bytes. The four parallel lines at the physical layer corresponds to the decoding of codeblock group of the same transport block. The first transport block includes the first IP packet entirely transmitted and forwarded to the PDN and a part of the second one. The third transport block includes the other part of the second packet and the third packet entirely. The second packet is reassembled at the RLC layer with the 2 TBs, that is made visible in Figure 4.2 with the vertical line at the `rlc.reas.am` segment. The delay introduced by the transmission of the 3 packets over 2 successive subframes is then 1 ms and depends on the achievable transport block size i.e. the achievable uplink throughput. A packet of size 8 kB is in this configuration divided into 6 IP packets,

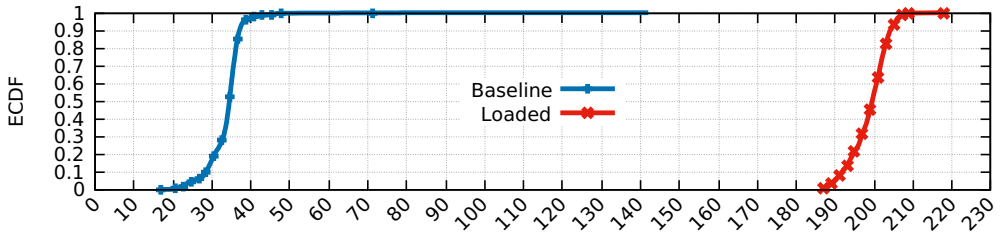



Figure 4.3 – ICMP ping RTT in a loaded cell at 85 % of network capacity in the downlink compared to baseline (ms).

transmitted over 4 radio subframes. The 3.5 ms of extra latency observed for 8 kB is then only the results of the transmission over multiple radio subframes.

**RTT in a loaded cell** In the BE paradigm, it is a very common situation where two different applications, with different traffic patterns share the same DRB i.e. the same network pipe. In this situation, the 2 flows are concurrent to access to the same shared resources which are PDCP entity ciphering functions, radio resources, etc. The worst situation for a mouse flow (i.e. low datarate flow) could encounter is the sharing of the same DRB with a long large flow (elephant-type flow). To emulate this situation, we continue to use ICMP ping at 5 packets per second for short time sensitive traffic, and we use shaped *UDP* traffic with a throughput equal to 85% of the cell capacity in the downlink (30 Mbps). Shaped traffic means that the maximum sending rate is controlled and packet are transmitted with a smoothed inter-departure time without burst. For UDP packet of size 1472 bytes, it means 2.5 packets arrives every subframe of 1 ms. Iperf tool is used to perform this experiment.

The downlink transmission profile shows a constant transmission of packets every radio subframe (i.e. every millisecond), as there are no other user in competition to radio resources access. As we do not reach the radio capacity, we only had 7 packets dropped (over 61665 UDP packets) following 2 HARQ retransmissions failed.

We plot in Figure 4.3 baseline latency and latency when the DRB is loaded. Ping RTT in function of packet size for an empty cell, corresponding to the previous situation with median RTT around 32 ms. When the concurrent flow is present, the median RTT has raised up to 200 ms, that is equivalent to a propagation delay of 40.000 km in an optical fiber.

Another remark when we observe the ECDF, points are less concentrated around the median, but extreme values (max and min) are less spread, 30 ms (for a median at 200 ms)

instead of 50 ms (for a median at 32 ms). The bandwidth is regularly given to the UE, the access time in the uplink is reduced. In the same time, the value are spread around the median because the small ping packet are in competition with larger UDP packets in the downlink, the difference of delay jitter introduced by the uplink is intensified by the downlink queue bottleneck.

In summary, this experiment illustrates the consequence on the RAN RTT of a persistent queue in the BS for a BE DRB. The problem could be tackled by avoiding intra-bearer resource sharing. Since by radio frame there are 29.8 packets of long flow and 0.05 ping packet, it is more interesting to organise a round-robin sharing between them with 2 DRBs (with an adapted QoS Class Identifier (QCI) for time sensitive flow). The scheduler multiplexes DRBs, not the UE or the flows. Slicing is a promising solution to differentiate flow types with reserved radio resources (see latency mitigation section 1.3.3).

### 4.1.2 Connection establishment, RTT and HTTP/S transfer

We evaluate in this section a real-world latency case with the HTTP downlink transmissions.

#### 4.1.2.1 Short HTTP transfer

The first experiment is a HTTP transfer to get small file of 10 kB. It shows the impact of the attachment delay and baseline RTT on the transfer duration since an important part of the transfer consists in awaiting responses during the connection establishment, RRC connection, TCP handshake and HTTP response (see Fig. D.A.4).

**Radio connection establishment** The attachment latency, consisting in RACH procedure, contention resolution and RRC procedure [30] (See Appendix A.A.1) delays takes at least 50 ms [3] and 76 ms in an average. In the 3GPP standard [89], control plane latency is defined as "*the time to move from a battery efficient state (e.g. `IDLE`) to the start of continuous data transfer (e.g. `ACTIVE`)*". It is equivalent to 2 full round-trips even with a low loaded mobile network core and an RRC establishment success of 100%, that is not negligible for short transfers in terms of latency and induced energy consumption.

3. `https://www.techplayon.com/lte-control-plane-and-user-latency-calculation-fdd/`

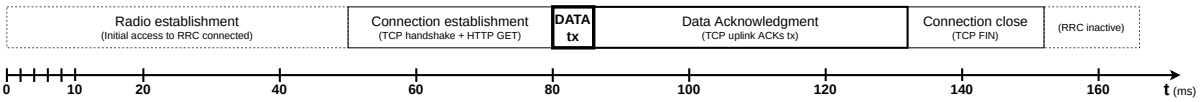



Figure 4.4 – Time components of the HTTP completion time.

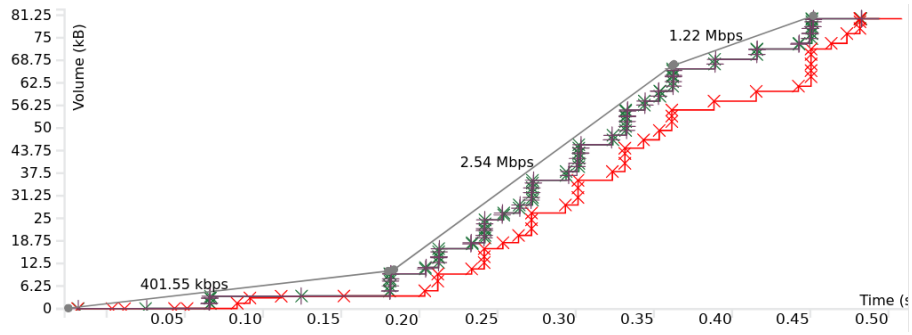
**Data transmission** HTTP lifetime connection is characterized by multiple handshakes and connection establishment procedures operated sequentially (see Appendix A.A.4 for the procedure time-sequence). Each packet exchange takes at least one RAN RTT. The lifetime of the transfer is depicted in Figure 4.4. A screen capture of all the packets referring to the HTTP transmission at the eNB's network interface and in eNB with LatSeq is attached in Appendix D.D.1. A description of the transmission is also provided. In total, on the 102 ms of the transmission duration, 30 ms are for the connection establishment (TCP SYN/SYN-ACK/ACK handshake and HTTP GET), 20 ms for the connection close (TCP FIN), 46 ms for the uplink TCP acknowledgments (ACK) and 6 ms for the effective downlink transmission of the data packets.

**Energy cost** This experiment shows that very short flows of few kilobytes are impacted by the initial access procedure delay and the RTT due to multiple packet exchanges for connection establishment and closing. While the transmission of the user data lasts for 6 ms, the UE modem should be turned on for 178 ms, with a cost in terms of energy.

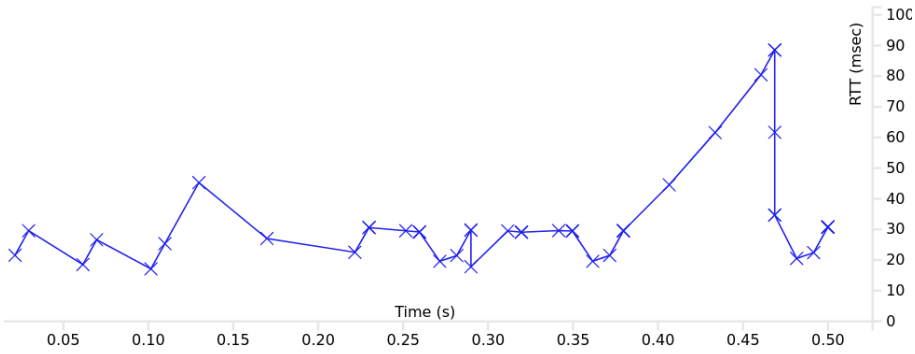
**Limitations** Latencies related to radio link errors are not included in the above experiment. MAC HARQ (§1.1.2.1) recovers most of the link errors, RLC ARQ recovers remaining MAC errors. 1 HARQ acknowledgment/retransmission cycle takes 8 ms (less in 5G), a significant latency compared to the RAN RTT.

DRX is an important mechanism for energy consumption saving by regularly turning off the radio channel. It follows typically a cycle alternating between sleeping and awaking [25, 199]. No DRX is configured in our experiments (see Table 2.2). The impact of DRX sleep cycle on latency could be important [200] with long sleep cycle to save battery [26].

In addition, the handover in mobility between two base stations is not evaluated despite the handover generates latency and head-of-line blocking as highlighted by [129] for mobile Virtual Reality (VR) traffic in LTE networks.




(a) TCP traffic profile. *Green: DL RAN ingress; Purple: UE DL reception; Red: UL RAN egress.*




(b) Associated RTT.

Figure 4.5 – Webpage HTTP transfer.

#### 4.1.2.2 Longer HTTP transfer

At 2022, webpages weight is commonly less than 4000 kB (equivalent to a 850 ms transfer at full bandwidth in our lab testbed) and constituted of 65 elements from different server locations [4]. The second experiment is an HTTP transfer using Transport Layer Security (TLS) encryption representing the typical downloading of a web resource of 80 kB length (i.e. 55 full MTU IP packets). The TLS encryption adds 2 RTTs of negotiation before the data transfer begins (see Client-Server hello in Appendix D.A.4). Contrary to the first experiment where all packets are transmitted in one burst, the TCP sender may act with several transmission/ACK rounds. Figure 4.5a represents the traffic profile for this HTTPS transmission. The firsts 10 kB of data are transmitted in one burst after the connection establishment that is achieved at second 0.15. Thereafter, the packets are transmitted as soon as the TCP sender transmit new bytes at the reception of an acknowledgment.

4. Live measurement campaign in Orange France network: `https://webview.orange.com` (visited in August 2022)

The experienced RTT at the beginning of the connection (during TCP handshake and connection establishment) is contained in a range of 20-35 ms in accordance with the measured baseline RTT in section 4.1.1. In the same time, the TCP sender seems not to be able to send enough packets to fill the bandwidth. In average, the achieved goodput is 2.5 Mbps, only 7% of the available radio capacity. It means that with a full usage of the radio capacity, the HTTP object could be transmitted in 17 ms and not in 270 ms as here.

The TCP time-sequence plotted in Figure 4.5a shows how packets are transmitted and acknowledged. The TCP sender increases its sending rate at the reception of a new ACK but it does not to reach in the time window of the transmission the network capacity, that is the primary objective of a TCP sender at the beginning of a connection. The transmission rate increase is also cut back after second 1.75 because of an increase of latency (Fig. 4.5b after second 0.40) from 20 to 90 ms. The uplink traffic profile in red in Fig. 4.5a explains the increase of the latency by a delayed transmission of ACK in the uplink, confirmed with associated LatSeq traces.

A webpage is typically constituted of dozen of HTTP elements of few kilobytes. The above experiment is repeated for each HTTP element to load a webpage (in parallel and in sequence according to element dependencies). Multiple HTTP transfers use a significant part of the available channel capacity (50% compared to 7% for one HTTP transfer) with an intra-bearer competition between flows with different RTT (depending on server location).

### 4.1.3 Exhibiting bufferbloat

*Bufferbloat* is an important issue of latency in cellular networks (see §1.3.2). The bufferbloat occurs where excessive buffer incurs long latency, substantial jitter and sub-optimal throughput. This section exhibits the bufferbloat and its impact on throughput and real-time response of applications. In particular, we show that uplink is more subject to bufferbloat than downlink.

**RRUL tests** The Realtime Response Under Load (RRUL) tests [5] is a common test in the bufferbloat research community. To perform this test, Flent [6] [182] was used. This test puts a pressure on the network with 4 TCP flows in downlink and 4 TCP in uplink

5. `https://www.bufferbloat.net/projects/codel/wiki/RRUL_test_suite/`
6. `https://flent.org/`

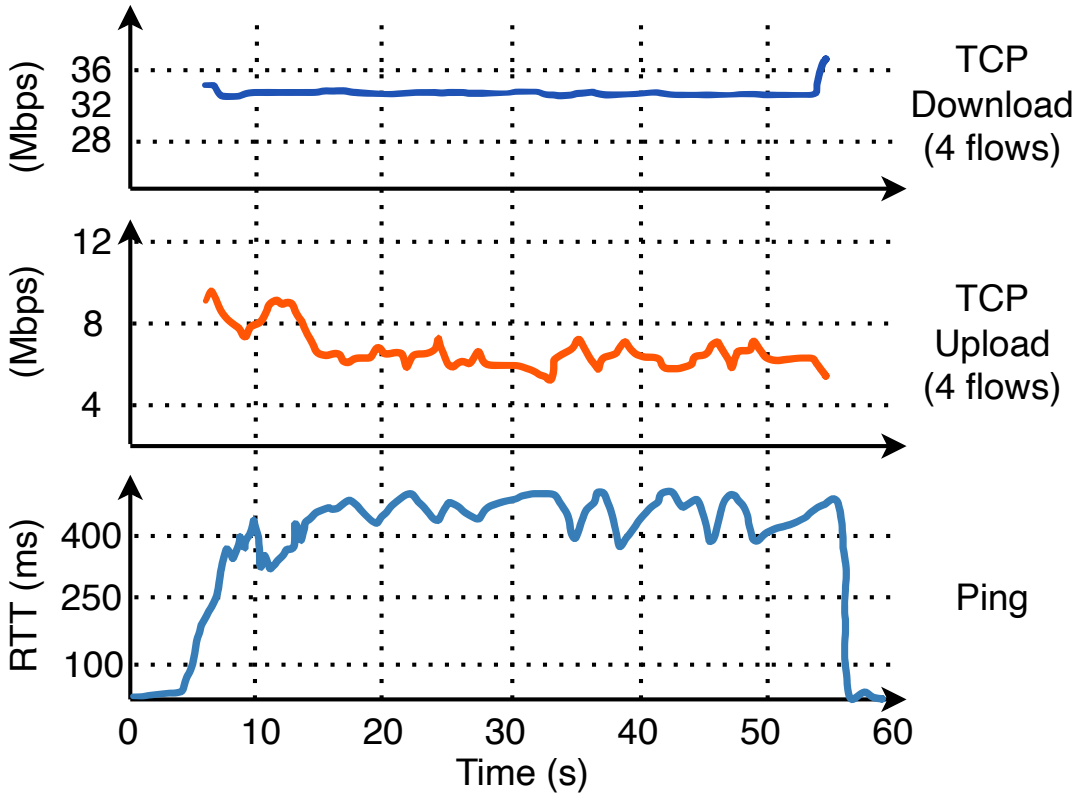
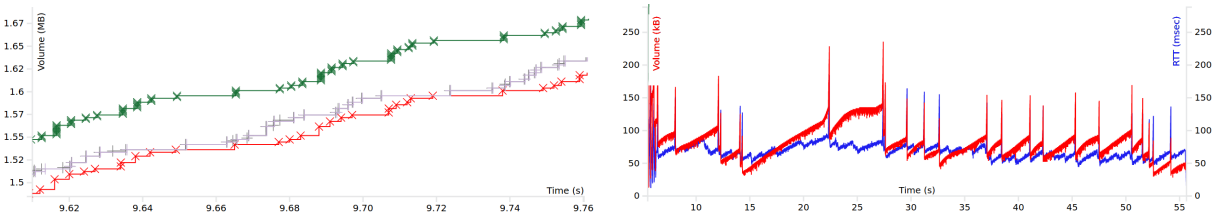



Figure 4.6 – Flent Real-time Response Time under Load experiment exhibiting bufferbloat effect.

in parallel, 1 ICMP ping for RTT measurement and 1 UDP flow during 50 seconds. The 8 bidirectional TCP flows emulate multiple long transfer in downlink and uplink while UDP flow emulates a real-time service. There is no network equipment using the QoS code points, then, flows are undifferentiated.

**Test Results** Figure 4.6 shows the downlink throughput, the uplink throughput and the latency measured by the real-time service flow. The sum of achieved datarate in the downlink is 33.28 Mbps (94 % of radio capacity used), 8.32 Mbps in the uplink (49 % of uplink radio capacity used) and the Jain fairness index of 0.9875 (1.0 is better). About 0.025 % of sent bytes have been considered as lost in the network by the TCP stack.

**TCP downlink flows** We select a TCP downlink flow in Figures 4.7. Figure 4.7a shows the traffic profile at different measurement points and Figure 4.7b shows the associated RTT. The RTT is in a range of $70 - 100$ ms. A finer analysis with traffic profiles and




(a) Associated traffic profiles. *Green: DL RAN ingress; Purple: UE DL reception; Red: UL RAN egress.*

(b) The corresponding RTT and bytes-in-flight. *Blue: RTT over time; Red: Bytes-in-flight*

Figure 4.7 – Flent RRUL downlink experiment analysis for a selected TCP flow.

LatSeq demonstrates that delays are concentrated on the downlink RLC queue.

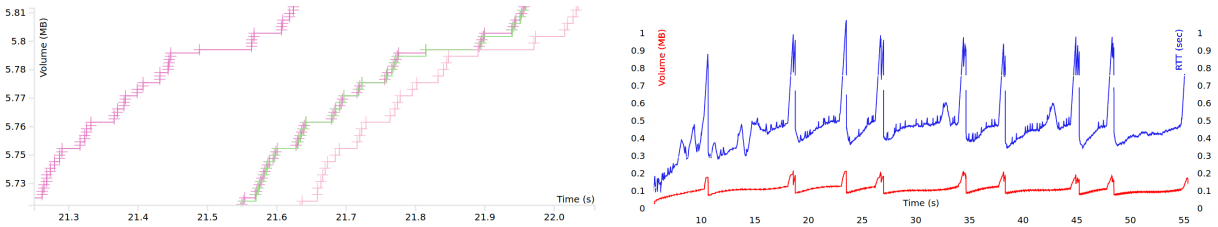
Since the UE is consistently scheduled over all the resource blocks in the downlink, the traffic profiles may suggest that the intermittent transmission is due a concurrency of access to resource blocks with other TCP flows. The concurrency between TCP flows in the downlink is visible by two elements:

— A larger queueing delay between the green and the purple line, meaning a larger number of packets in the RLC transmission buffer.

— In the downlink interface (purple line), the packets are not yet transmitted regularly every millisecond but transmitted without a clear pattern as the radio resources are shared between flows without differentiation.

ACK are quickly generated after the reception of a data packet and then are buffered in the UE's uplink transmission buffer. The total standing RLC downlink queue is about 400 kB and in average 90 kB for this TCP flow, showing the fair share of the downlink buffer. The Bytes-in-Flight (BIF) shows the convergence of the TCP sending rate towards the fair share use of the link with the other TCP flows. Remember that this situation is very particular since there are no delay between RAN and server, there are no competition with other connected user to schedule, and the network interface capacity does not vary in time (i.e. the MCS remains high and constant). However, this experiment in the downlink shows how the downlink queue builds up with concurrent TCP connections.

**TCP uplink flows** In the uplink, data packets are carried by the uplink segment while small acknowledgment packets are traversing downlink segment. The achieved datarate is only 6.45 Mbps (less than 36.7 % of radio resources used). The average throughput is 1.61 Mbps with a fairness index of 0.9858 among TCP flows. About 1 % of bytes are considered as lost by the TCP sending stack. The figure of uplink RTT (Fig. 4.8b) shows




(a) Associated traffic profiles. *Magenta: UE UL egress; Brown: UL RAN egress; Ping: UE UL ingress.*

(b) The corresponding RTT and bytes-in-flight. *Blue: RTT over time; Red: Bytes-in-flight*

Figure 4.8 – Flent RRUL uplink experiment analysis for a selected TCP flow.

that the retransmissions are triggered because of an RTT over to 600 ms. Compared to the downlink, the uplink flows show a much higher RTT of around 450 ms. For both uplink and downlink segments, the delay is more important for TCP uplink flows than for downlink. The uplink segment represents the major part of the RTT, with a contribution of 300 ms visible in Figure 4.8a between the magenta and brown curves. The downlink segment delays acknowledgment packets of 100 ms (between brown and pink). A comprehensive study on-device uplink bufferbloat is provided by Guo et al. (2016) in [22].

The uplink and downlink latency different results reveals fundamental asymmetries between radio downlink and uplink side. This test exhibits two of them about the scheduling method:

— The difference of *achievable throughput* is less important in the uplink than in the downlink for a given bandwidth. Indeed, because of the limited uplink emitting power, and the limited computation capabilities, the modulation coding scheme in uplink is limited to small value.

— The difference of *radio resource management* inherent to the centralized architecture of the cell. The scheduling of uplink transmissions is operated by the BS at the reception of SR and BSR sent by the UE. We observed radio frame where the UE does not request uplink radio resources and then, BS does not grant bytes to it, increasing the bufferbloat effect in the uplink.

## 4.2 Uplink Segment as a Source of Latency and Jitter

This section presents in detail experiments and findings related to the RAN uplink-channel latency. Especially, we demonstrate the importance of uplink radio access and radio resource allocation scheme on the uplink latency characteristics.

### 4.2.1 Uplink transfer experiment

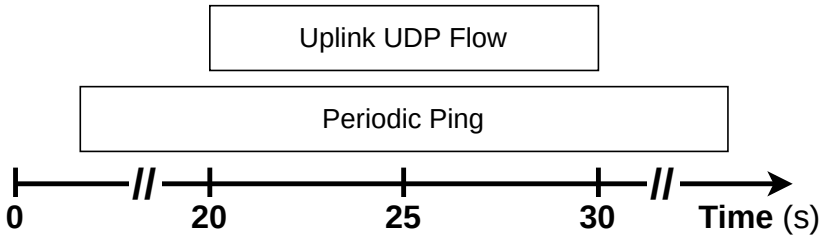


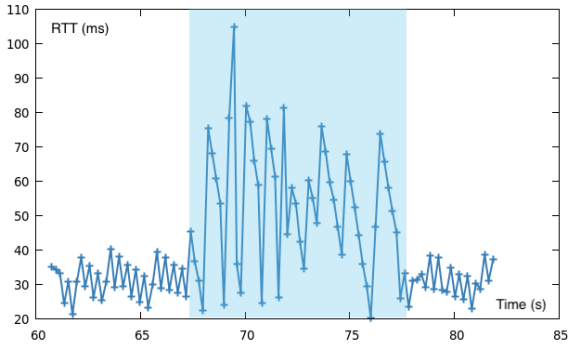

Figure 4.9 – Uplink experiment transfer scenario

#### 4.2.1.1 Experimental setup

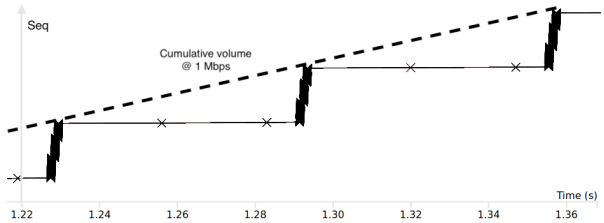
We employ a long UDP flow (10 seconds transmission) at a datarate of 1 Mbps in the uplink. The UE generates traffic and the server receives UDP packets. The datarate is chosen to generate a continuous flow of packets without reaching the bottleneck of radio capacity. Packet length is set to have the same size as a TCP pure acknowledgments packets (= 24 bytes of UDP payload, 80 bytes in total). That is small packets compared to Ethernet MTU size packets used in the downlink for data transfer. Notice that small packets ($<$ 200 bytes) in the uplink are very common among applications for instance TCP downlink acknowledgments, cloud gaming applications, remote control. This traffic generation configuration ensures to generate at least one packet every millisecond, maintaining a non-zero transmission buffer at each TTI. The RTT is measured all along the experiment with an ICMP ping as illustrated in Figure 4.9.

#### 4.2.1.2 Results

**Latency** Main results are presented in Figures 4.10. The variation of RTT before, during, and after the generation of UDP uplink traffic is plotted in Fig. 4.10a. Without the UDP traffic, the RTT varies from 20 to 40 ms as we already pointed out in Fig. 4.1. When the UDP traffic is generated, the average RTT increases quickly to 50 ms with an




(a) Ping RTT (ms) measurement over time (s)




(b) Associated uplink traffic profile


(c) Associated BSR (green) and SR (brown) signalling

Figure 4.10 – Results of baseline uplink transmission at 1 Mbps experiment.

important jitter between ping measurements from 20 to 80 ms with a maximum to 110 ms. The significant variation of RTT from one measurement to another looks surprising because the UDP packets are generated at a Constant BitRate (CBR) and thus, the increase of RTT must be constant.

**Traffic profile** The uplink traffic in Fig. 4.10b (zoom on a time window of 240 ms) exhibits a "stepped" transmission with burst of 3 TTIs length every 64 ms carrying the most part of the traffic volume interspersed with less regular small transmissions carrying one or two packets. The consequence of that is an important variation of the RTT, exacerbated by the transmission in "bursts". We define the burst of transmission as the transmission of at least three data packets over one or more consecutive TTIs. On the traffic profile presented, there are 3 bursts of transmission carrying in average 60 packets over 3 TTIs.

**Reported buffer** The reported buffer size by BSR [7] (see Fig. 4.10c for BSR signal) is almost constantly the same with index 42 (= [ 5476 , 6411 ] bytes) at least every 64 ms. The BSR value in addition to report a transmission buffer size of around 6000 bytes also shows the stability in time between buffer filling and buffer emptying. However, the

7. §1.1.3.2 for a BSR signal description

added queueing delay measured of ≈ 25 ms is not coherent with the buffer length and the radio capacity because a quick calculation suggests that the queueing delay should be $6000 \times 8/17.10^6 = 3$ ms.

#### 4.2.1.3 Conclusion

This experiment arises 3 questions about uplink-channel transmission: 1) *An important uplink latency*; 2) *An important jitter* from a packet to another; 3) *A transmission per burst* rather than a continuous transmission.

The latency (1) is the consequence of packets transmission delayed into a burst of transmission. The jitter (2) is also due to the transmission in bursts of packets arriving at different moments but also due to small packet transmission between bursts. The bursts of transmission (3) come just after the reception of a BSR with a great correlation in time with a delay shift of 8 ms. It has to be noted that the burst size corresponds to the reported buffer length. The small transmissions do not appear to be correlated to the reception of a BSR and they are weakly correlated to the reception of a SR.

The hypothesis is that the transmission pattern, cause of the observed latency and jitter, is a consequence of the radio resource allocation process. Indeed, there are no scheduling prioritization between UEs, air interface capacity variability or radio link retransmissions that could generate this kind of bursty transmissions. We remade the experiment with a lower constant MCS and we got the same result on the traffic pattern with bursts of transmission that lasts over more TTIs.

### 4.2.2 Uplink latency and jitter in a commercial RAN

We seek to reproduce the previous experiment in a commercial RAN to validate previous conclusions in an other LTE system.

#### 4.2.2.1 Commercial Radio Access Network setup

In this setup, the radio access network is considered as an object of study on which we perform experiments under different conditions. Radio access network configurations are the same as those used by other users and terminal is immobile so as not to add uncontrolled parameters due to propagation condition variations. The parameter of these experiments are the type of traffic relaying by the cellular network and the load of the cell. End-to-end latency comprises the RAN latency but also on-terminal latency, core

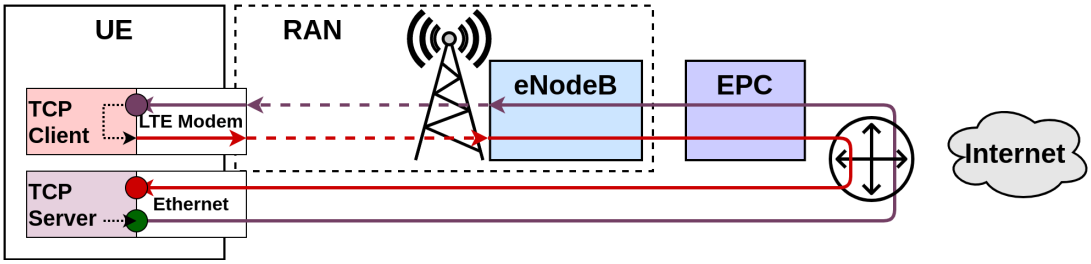



Figure 4.11 – Commercial network experimental setup.

network latency and internet latency. The presented setup is designed to isolate the RAN latency as much as possible.

| **Configuration** | **Value** |
|---|---|
| RAT | LTE |
| Band | 3 |
| Frequency | 1800 MHz |
| Mode | FDD |
| Bandwidth | 10 MHz (50 PRBs) |
| MIMO | $2 \times 2$ |
| Base Station model | Huawei |
| Cellular modem model | Huawei E5377 |
| Radio Bearer QoS | Best-Effort |

Table 4.3 – Commercial network testbed configurations.

The experiment takes place in an Orange's cellular network the configuration in Table 4.3. Packets transfer are made between a client and a server but both are located in the same device (*i.e. the UE*), which creates a local breakout as shown in Fig. 4.11. Client and server generate UDP and/or TCP traffics according to the type of service for which we would observe experienced latency. The client is connected to the cellular network interface. The server application uses the Ethernet interface. Thus, network captures and logs for client and server shares the same clock with a perfect synchronization, easing the problem of end-to-end latency analysis. Also, we avoid the problem of uncontrolled internet delays by connecting the server at the nearest point of the core network. We were not able to verify the information and puts it as hypothesis, that latency introduced by the core network is negligible and it does not constitute a latency bottleneck. Thus, the end-to-end latency measurement is equivalent to the Radio Access Network (RAN) Round-Time Trip (RTT).

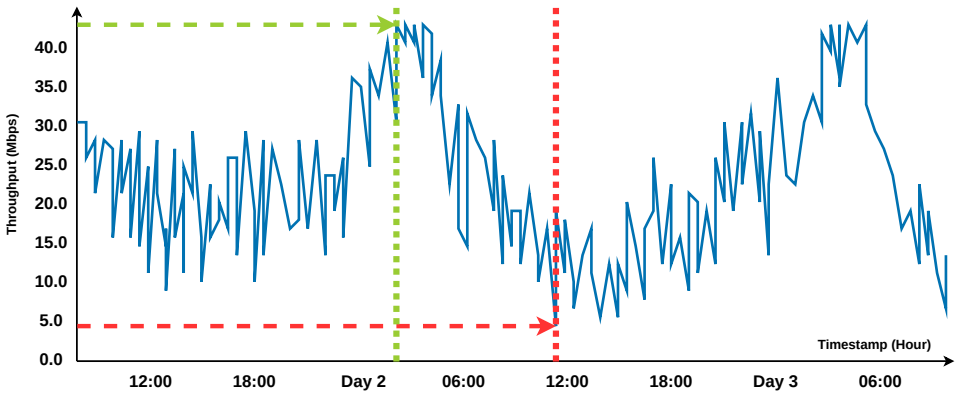



Figure 4.12 – Evolution of the achieved throughput in the cell all along the day.

The band 3 is used, corresponding to a frequency of 1800 MHz (FDD, uplink: $1710-1785$ MHz, downlink: $1805-1880$ MHz) associated to a bandwidth of 10 MHz. The antenna uses the MIMO technology $2\times2$. A quick calculation gives a maximum achievable throughput for the cell of 75 Mbps in the downlink and 25 Mbps in the uplink. Huawei is the vendor for both the base station and the cellular modem receiver. The terminal is in a fixed location and in good radio condition i.e. the SNR is high and constant. Such radio conditions minimize latency due to radio layer retransmissions and concentrate latency study on access and scheduling latency. The cellular network interface is made using a Huawei E5377 mobile hotspot [8] with a theoretical maximum throughput of 150 Mbps. The server uses a gigabyte interface. The operating system of the host machine on which client and server are installed is a Linux kernel 4. Network captures are made on the 2 network interfaces as indicated in Fig. 4.11.

The cell serves regular customers in BE way, with a fair sharing of the radio resources. All along the day and mobility events, the load of the cell is varying according to the customer usage of the network. Fig. 4.12 represents the achieved throughput for the testing terminal all along the day. The achieved throughput is the consequence of the instantaneous cell load and the sharing of the radio resources among users. The highest throughput is achieved during the night and the lowest during the afternoon when the number of connected users is the most numerous and the most active. The plot shows a limit of throughput at 43 Mbps during some measurements at night when the cell is almost empty. We use as hypothesis that 43 Mbps is the maximum achievable throughput in the given setup.

8. `https://assistance.orange.fr/equipement/cles-3g-et-dominos/huawei-domino-4g-e5377`

For the following observations, we perform 2 UDP uplink transfers at 1 Mbps (< 43 Mbps, the radio channel capacity) at the 2 moments pointed with arrows in Fig. 4.12. The first when the network is empty at night and one at noon when the cell is busy.

#### 4.2.2.2 Observations on the uplink traffic profile

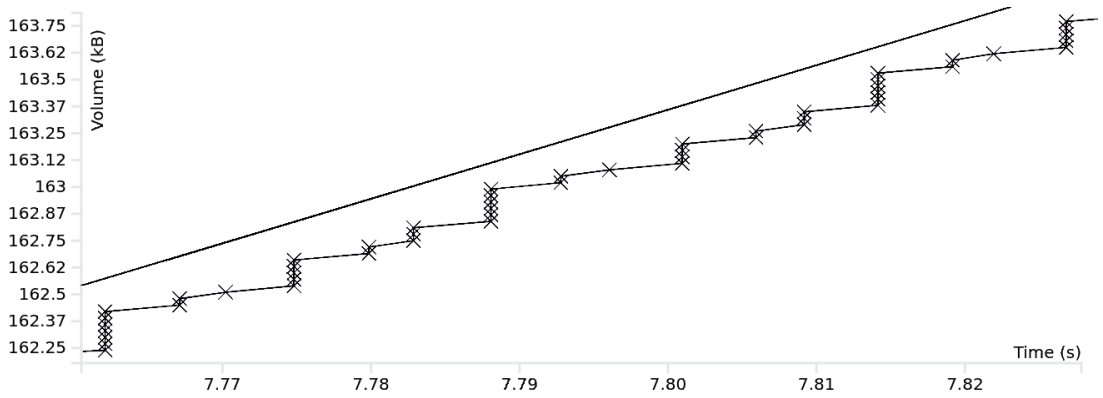



Figure 4.13 – Uplink traffic profile associated to a transmission at 1 Mbps in empty commercial network.

**Empty cell case** The traffic profile at a small timescale window of 70 ms (see Fig. 4.13) exhibits a similar uplink traffic pattern as we found in section 4.2.1: 1) An important uplink latency for some packets; 2) An important jitter from one packet to another, especially between packet transmitted "alone" and those transmitted in a burst; 3) The presence of regular burst of transmission preceded 5 ms earlier by an uplink transmission.

In this experiment, the scheduler has no reasons to delay a grant requested by the device while there are PRBs left in a TTI (generated datarate inferior to the cell capacity) and there are no competition with another device.

The duration between 2 bursts of transmission is 13 ms in 90% of cases. With the same reasoning as with the experiment on the OAI testbed, a BSR transmission should be associated to a burst of transmission. The possible BSR periodicity around 13 ms is 8 or 16 ms [37]. The time between a TB transmission and a burst is 5 ms, that is the minimum achievable latency in LTE-A for the SR-to-Grant-to-Transmission round.

**Loaded cell** In a loaded cell where the UE is in competition with other UEs the traffic profile in Fig. 4.14 shows the same pattern with bursts and small transmissions with some irregularities in the interval between bursts. Also, the pattern between burst is not yet

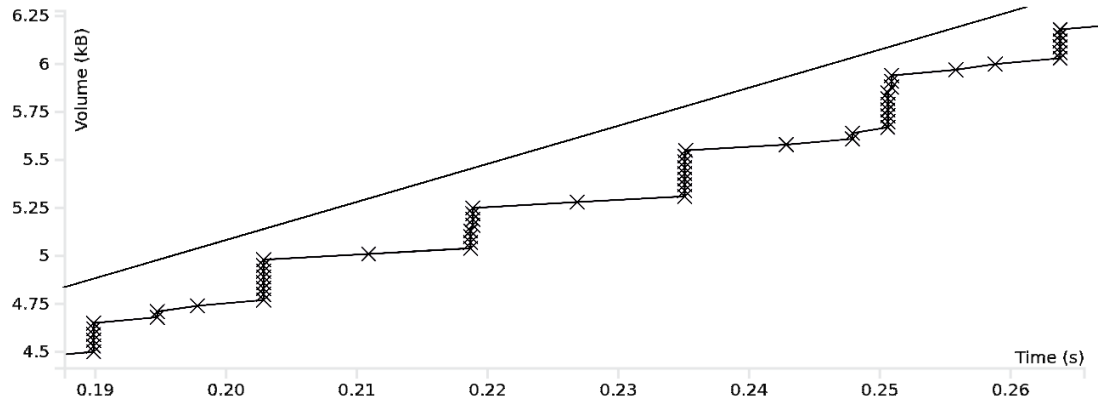



Figure 4.14 – Uplink traffic profile associated to a transmission at 1 Mbps in loaded commercial network.

clear. The new variables introduced by the resource sharing aspect are : 1) the delay between the explicit scheduling request (BSR or SR) and the grant; 2) the size of grants. The competition between UEs to access to a given TTI induces a variable delay that is not yet equal to 4 ms but greater. The competition between selected UEs for a given TTI reduces the achievable instantaneous throughput (explaining why some bursts of transmission span over 2 or 3 consecutive TTIs instead of one previously). The increased latency and jitter due to resources sharing is expected.

We show with these experiments that the traffic pattern even in the best conditions is not optimal concerning latency, and jitter, and it depends on specific RAN configurations.

### 4.2.3 Analysis of the uplink radio resource allocation scheme

#### 4.2.3.1 Static code analysis of the OAI base station stack

This section reviews the implementation of the standards that is made in the OAI RAN project [201]. It completes the pure reading of the 3GPP standards in paragraph 1.1.3.2.

At the reception of a SR, the BS sets for the connected UE the flag `ul_SR` and puts the UE in an active mode (Listing D.D.2). In response, the uplink dynamic scheduling algorithm that is executed each TTI, will assign by default 3 Resource Blocks (RBs) to the UE as soon as the BS as the estimated buffer `B` ($\hat{Q}$) is equal to 0. The code associated to these instructions of the scheduler is in appendix, Listing D.D.2. Moreover, the by-default MCS of 10, which is the highest MCS index to have a transmission in Quadrature Phase Shift Keying (QPSK), gives a TBS of 63 bytes [9]. This value of MCS is modified

9. In the srsLTE implementation[202], the BS assigns by default 64 bytes instead of PRBs directly at

later if a previous CQI has been calculated for this UE. The BS unsets the SR flag as soon as RB are assigned to a user. When a `BSR` is received at slot $n$, the buffer length estimation is simply updated to the value $Q^{\text{bsr}^+}$ reported according to the BSR index table (Listing D.D.3) such as:

$$\hat{Q}_n = Q^{\text{bsr}^+} \tag{4.1}$$

At the reception of a RLC PDU after TB decoding (Listing D.D.4), the estimated buffer $\hat{Q}_n$ is decreased by the size $L$ of the PDU such as:

$$\hat{Q}_n = [\hat{Q}_{n-1} - L]^+ \tag{4.2}$$

The selection of UE $\mathcal{U}$ to be scheduled in OAI depends on SR, BSR and an inactivity period. The code associated to the function `UE_is_to_be_scheduled(...)` that performs this selection is given in appendix, Listing D.D.5. The conditions to schedule a UE could be summarized as:

— The estimated UE transmission buffer is not empty (related to BSR-based estimation)
— A SR has been received, and no grants has been generated yet
— The UE is connected and active but no activities have been detected since 2 radio frame (= 19 ms)

After that, the scheduler shares PRBs $\mathcal{K}$ between $\mathcal{U}$ according to its algorithm and the scheduling strategy §1.1.3.2. At the end, a grant is issued with the number of resource block $K_i$ to the UE $i$, the resource block map, the MCS to use, the transmission power and other control flags (Listing D.7.6).

#### 4.2.3.2 Illustration with an uplink traffic profile

Figure 4.15 illustrates the consequence of the radio resource allocation process implementation on the uplink traffic profile. At the bottom of the figure, the generation of small packets (less one hundred bytes-length) at the UE side which are accumulating in the transmission buffer. At the middle, the air interface transmission and at the top, the egress interface of the base station with the packets transmitted to the core network. Applying the code analysis above, we get the following transmission events:

the reception of a SR

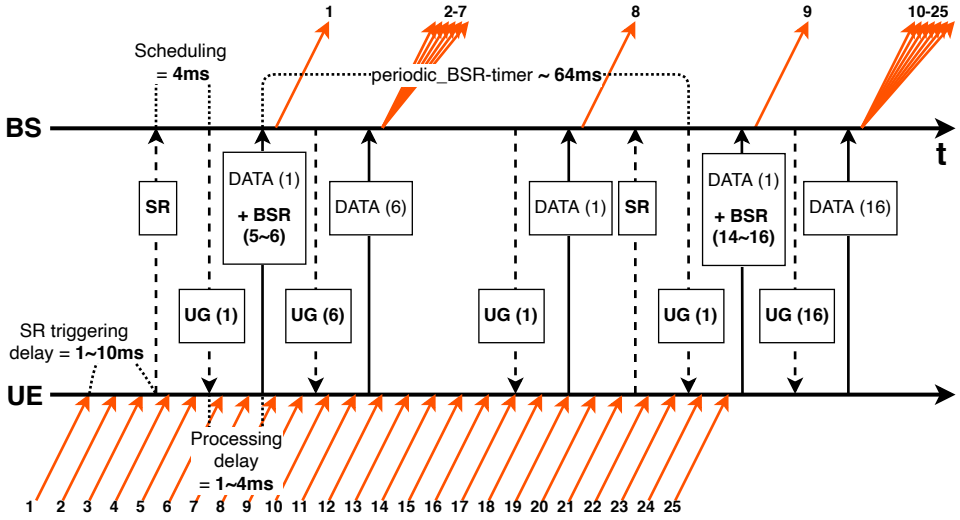



Figure 4.15 – Grant-based uplink access dynamic with explicit SR and BSR.

1. When the first packet arrives in the UE transmission buffer, a SR is triggered and sent at the next SR transmission opportunity.
2. The BS in response sends a UG that corresponds to a TBS of 3 PRBs, corresponding to 68 bytes $\sim$ one packet (`UG (1)`).
3. The UE uses this opportunity to transmit BSR and completes the TB with some user data. The packet 1 is small enough to be transmitted entirely within the TB. The OWD experienced by the packet 1 is then about 13 ms.
4. The BSR reports the size of the buffer containing packets number 2 to 8. It is indicated with `BSR(5` $\sim$ `6)` on the figure where the BSR reports in reality a range of bytes corresponding to 5 or 6 packets in our example.
5. An UG is sent accordingly that incurs the transmission of these packets in a burst of 6 packets 8 ms later.
6. For an unknown reason with certain model of smartphone in same radio context, the UE stops to send SR for a while, such as it has a DRX configured, but we verify that our network core does not configure a DRX. We think about an energy consumption manager implemented in the modem.
7. However, because the UE is RRC connected, after an `inactivity_timer` period the base station "wake up" the UE with a UG as if the base station received a SR. The UG size should not correspond to the transmission needs since no BSR has reported its size yet.

8. In response, the UE transmits data but neither a periodic BSR because the timer has not expire yet, nor a regular BSR because it is the low priority default logical channel nor a padding BSR because there are data in the buffer to transmit.

9. Later, a periodic BSR is transmitted at the expiration of the periodic BSR timer `periodic_BSR-timer`. Afterwards, a burst of transmission occurs with packets numbered from 11 to 25. The uplink OWD experienced by the packet 11 is about 55 ms to 10 ms for the packet 25 for the same data burst.

Hence, that is not a mystery to observe a burst of transmission 8 ms after a small UG, which betrays the reception of a BSR and an update of the estimated buffer. In fact, the inter-arrival of the bursts is almost constant at 64 ms, corresponding to the BSR periodic timer configured of 64 ms. At the update of the buffer status, the BS allocates resources in consequence to empty the buffer at the maximum datarate. Given the radio capacity, it takes 3 TTIs to empty the buffer built up during 64 ms. With a 20 MHz bandwidth with the highest MCS it could take only one 1 TTI to empty entirely the buffer. Conversely, with a 1.4 MHz bandwidth (= 6 PRBs) with a middle value MCS, it would take at least 50 TTIs. In last situation, the effect of jitter due to BSR reception should be vanished.

An unexpected consequence of the default allocation following a scheduling request is the uplink delay jitter generated in uplink for small packet flows. Indeed, in the presented mechanism above, a big packet of 1500 should be segmented at the RLC layer level without to be completed with two or three small transmissions but reassembled at the burst reception with the remainder of the packet, reducing the interpacket-jitter but with a cost in latency for this unique packet.

The same analysis of the source code could be made on the srsLTE [10][202] project and leads to the same conclusions about the radio resource allocation according to the reception of SR and the buffer status estimation from BSR.

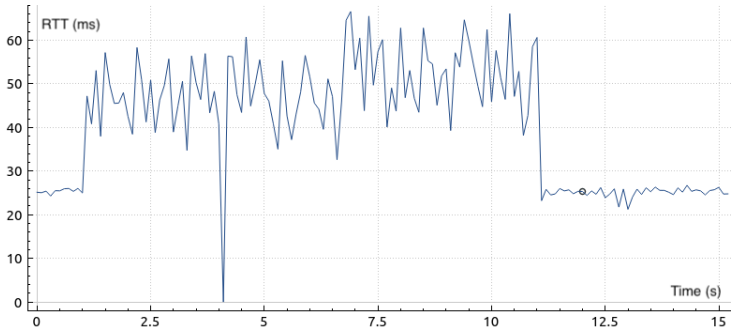
Patriciello et al. [43] in their implementation of the 5G-NR RAN precises also: "*Note that the first scheduling assignment [following a SR] is blind since the gNB does not know the buffer size at the UE yet. In this regard, since this is implementation-specific, we assume that the first scheduling opportunity consists of the minimum amount of OFDM symbols that permits at least a* 4 *bytes transmission.*". Thus, the same transmission pattern should be observed in their NS-3 5G simulator.

10. Github.com/srsRAN : `https://github.com/srsRAN/srsRAN`

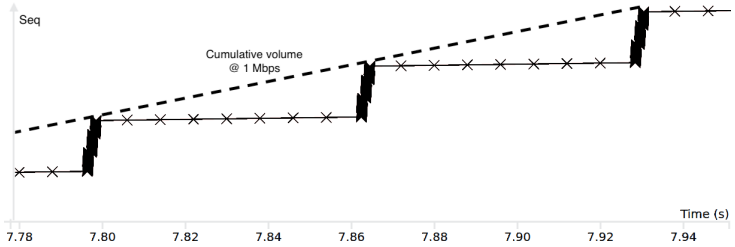
### 4.2.4 Influence of RAN configurations

In the previous section, we described the mechanism of allocation for the default RAN configuration and a CBR traffic. This section explores the influence of configurations, SR periodicity, BSR triggering and bandwidth allocation on the uplink traffic profile and uplink latency.

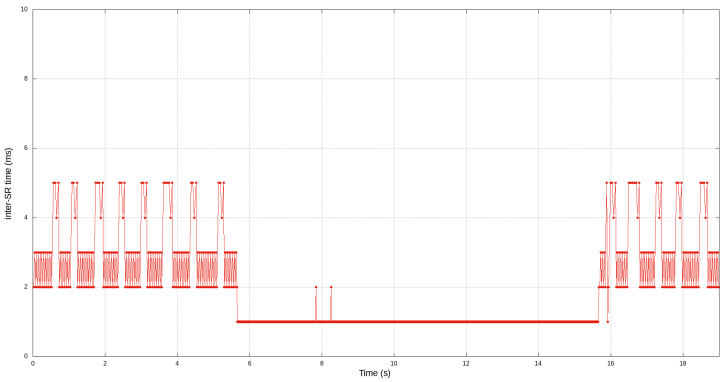
#### 4.2.4.1 Scheduling Request periodicity




(a) Ping RTT measurement (in ms)




(b) Associated uplink traffic profile




(c) Inter-SR time (ms)

Figure 4.16 – Experimental results with a Scheduling Request periodicity at 1 ms and an uplink transmission at 1 Mbps.

With the `sr-ConfigIndex` of 157 instead of 5, a SR could be sent every subframe (i.e. every 1 ms). This configuration avoids the necessity for the UE of waiting the next opportunity, eliminating the SR triggering delay. According to [129]-§6.1, it is 80% of packets that waits for a SR. First, we cannot see any differences for the transfer phase because, this phase is mostly controlled by the BSR reception. Then, a shorter SR periodicity is not used in this phase to reduce latency. More importantly, in the ping phase. The SR is triggered as soon as a new packet arrival as shown in Figure 4.16c where the inter-SR transmission during the transfer is equal to 1 ms (the mechanisms in the modem to trigger SR are not *a priori* known, it depends on the founder implementation), generating huge amount of SR. The BS answers to the scheduling request with a UG of fixed size of 3 PRBs. Because the UE will receive the UG through the Physical Downlink Control Channel (PDCCH) only 4 ms later (common scheduling delay in LTE), the UE continues to send SR until the limit of 4 (defined by the `dsr-TransMax` configuration).

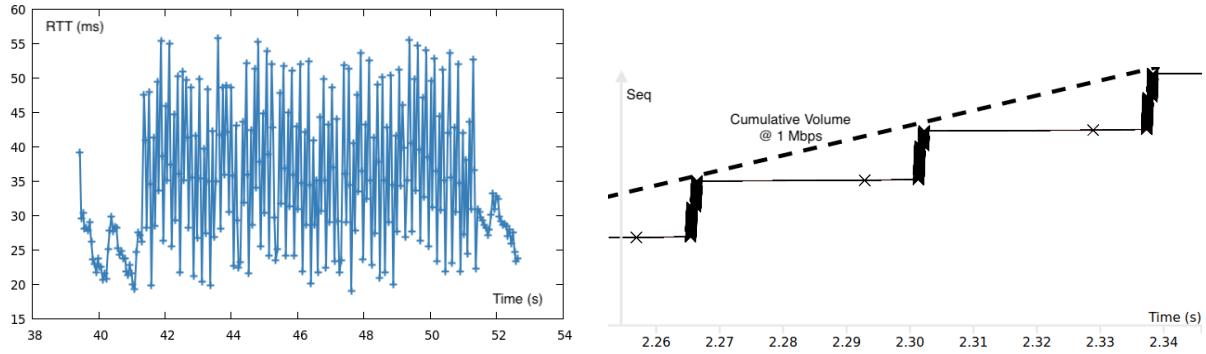
The RAN RTT is clearly reduced by reducing the SR transmission opportunity periodicity from 31 ms to 25 ms (see Fig. 4.16a and more importantly, more stable with a standard deviation of 1 ms contrary to 5 ms with default 10 ms SRs When the RAN is loaded, the RTT is reduced of 5 ms while the effect on jitter is less visible.

The traffic profile in Figure 4.16b shows more "little transmissions" between bursts of transmission. It confirms that without a BSR, the BS reacts in a conservative way and only assigns the amount of PRBs necessary to empty the estimated transmission buffer.

The scheduling round time (SR-to-transmission time) is 5 ms at minimum in LTE-A and commonly 8 ms. This value should tend to the millisecond in 5G with proper radio configurations of "K"s timings. The SR periodicity is typically to 10 or 20 ms according to [203],[129]-Table 7, values also used in OAI and srsLTE [202, 204]. But it does not constitute on itself a solution to reduce uplink latency because it does not reduce bursty transmissions while it costs in terms of energy for the UE to transmit.

#### 4.2.4.2 Buffer Status Reporting periodicity

We set the BSR periodic timer at the DRB establishment to 8 ms instead of 64 ms. Thus, BSR timer is expired at every SR (SR transmission opportunity periodicity to 10 ms). The impact of this configuration is clearly visible on the uplink traffic profile in Figure 4.17a. Every 10 or every 27 ms if there are no SR received, a small UG with a pre-allocated number of PRBs is issued. In response, all TBs includes a periodic BSR MAC Control Element (MAC CE) because the timer has expired. 8 ms later a burst of




(a) Ping RTT measurement (in ms)

(b) Associated uplink traffic profile

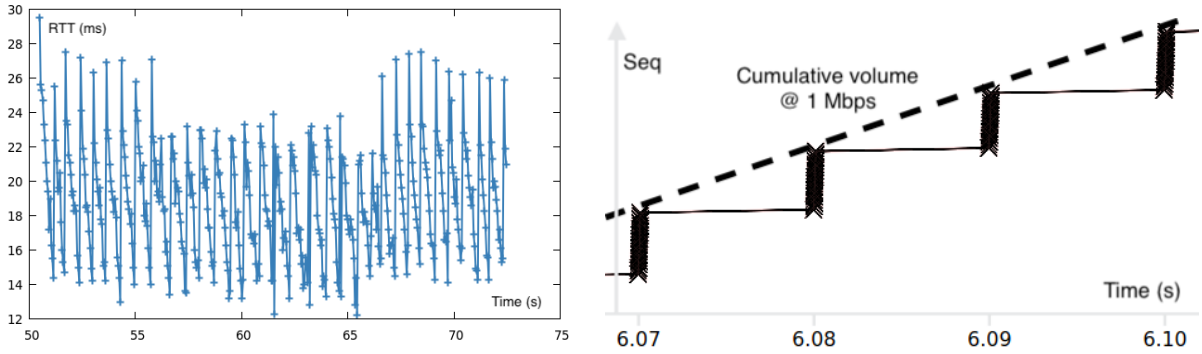
Figure 4.17 – Experimental results with a Buffer Status Reporting periodicity at 8 ms and an uplink transmission at 1 Mbps.

transmission corresponding to the reported BSR is visible. Thus, the transmission bursts occur more regularly, every 18 or 35 ms reducing the average RTT from 55 to 37 (see Fig. 4.17b) and the jitter from one packet to another from 50 ms to 30 ms in relation with traffic profile.

This experiment shows the significant role of the BSR reporting on uplink dynamic scheduling and then, on the latency characteristic. However, reducing BSR periodicity has a cost in terms of control overhead. The regular BSR is a control element of $1-4$ bytes-length in the MAC TB. A BSR transmitted every 64 ms as it is configured in OAI costs for a pool of 20 connected UEs 10 kbps. With a configuration of 8 ms in a loaded cell with up to 100 devices, the cost of BSR increases to 400 kbps only to reports buffer status. In the same time, it has a cost in terms of processing for both UE and BS for an assumed limited benefit. That is a trade-off between the knowledge accuracy we want at the base station for the scheduling and the costs we want to pay in terms of resource blocks and processing.

#### 4.2.4.3 Allocate all the bandwidth to UE at an explicit scheduling request

With this third experiment (Figure 4.18), we modified the uplink scheduler to allocate all the bandwidth of a slot to the UE each time it is scheduled. By then, the UE always receives the maximum radio resource capacity on a given slot. That is an unrealistic in practice, but it helps to understand what happens if the scheduler always guest the buffer status without reporting delay, inaccuracy and periodicity. The UDP uplink traffic profile depicted in Fig. 4.18b shows a clear regularity of the transmissions with small burst of




(a) Ping RTT measurement (in ms)

(b) Associated uplink traffic profile

Figure 4.18 – Experimental results for a full bandwidth allocation to an explicit scheduling request and an uplink transmission at 1 Mbps.

transmission every 10 ms without "pauses" or small transmission. The UE requests radio resources every 10 ms according to the SR periodicity and receive an UG that allocates all the PRBs of the slot to it. A slot with the given MCS and bandwidth could carry 2385 bytes. The traffic generates at the RLC level 1050 bytes of data every 10 ms (1 Mbps /8/1000 + header overheads). Thus, only 44% of radio resources are used for active slot. With a 20 MHz bandwidth it would be 20%.

Contrary to the previous experiments, we observe many SR triggering, every 10 ms in fact. The cycle of 27 ms due to eNB scheduler is not yet visible. Periodic BSR continues to be transmitted with a value of 0 every 70 ms ($> 64$). The question that comes is why a SR is triggered at every SR transmission opportunity whereas it was not the case in the baseline experiment. Our hypothesis is that, because BS allocates every time a large band (and not few PRBs), the UE modem never enters a "sleep" mode for battery efficiency, in other words, the high rate uplink keeps the modem on when a SR transmission opportunity occurs.

This experiment demonstrates the existence of a "sleep" mode for uplink interface, but we could not pursue the investigation since it is an internal modem mechanisms. We repeated the experiment with 2 different models of smartphones and observed the same phenomena that in the baseline case we have few SR compared to this fullband allocation experiment.

Concerning the latency measured with the ping RTT (in Fig.4.18a), the latency and the jitter is clearly decreased before and during the uplink traffic generation. The minimum RTT is reduced to 12 ms that is close to the minimum achievable in LTE that is 10.5

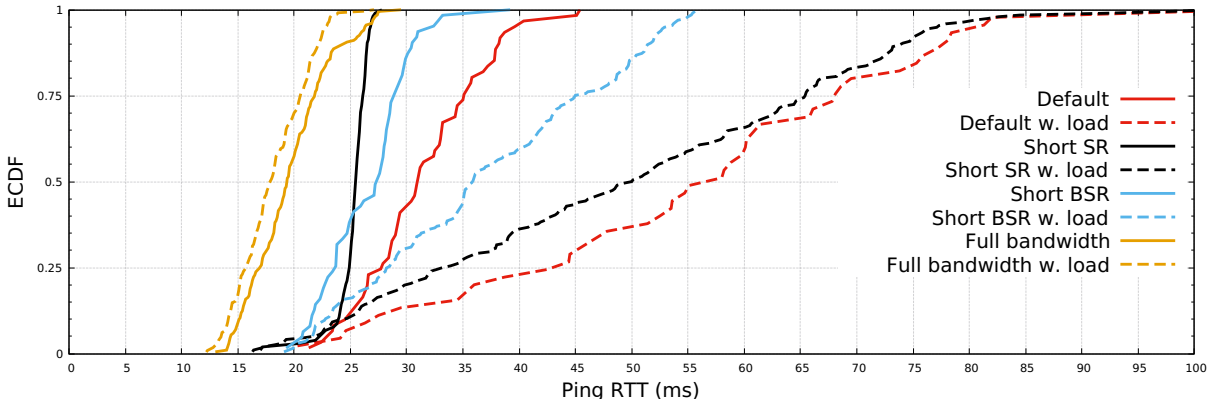



Figure 4.19 – ECDF of latency per RAN configuration. *Default : SR*=10 *ms, BSR*=64 *ms, PF scheduler; Short SR : SR*=1 *ms; Short BSR : BSR*=8 *ms.*

ms. The mean RTT is about 21 ms instead of 30 with a jitter of 14 ms without the uplink traffic. More interestingly when the generation of the uplink traffic, the mean RTT drops to 18 ms with a lower jitter of 10 ms that looks counter-intuitive. In fact, the generation of the uplink traffic increases the number of transmission opportunities with a regularly generated SR without filling all the TB. Then the ping RTT benefits from the regularly allocated TTI without being in competition to be carried in the transport block. Conversely, when the ping is the only traffic, it should trigger the SR and wait for the grant since the inter-departure of the ping is far more than 10 ms.

This experiment shows the importance of a short regular scheduling with the appropriate amount of PRBs to tend towards low latency and low jitter. But this amount of PRBs to allocate is not easy to determine under the resource usage constraint and partial knowledge of transmission buffer length.

#### 4.2.4.4 Summary

In this section we investigated the influence of RAN configurations, i.e. the SR periodicity configuration, the periodic BSR configuration and the scheduler response to a scheduling request, on the uplink traffic profile and the RAN RTT. ECDF of RTTs obtained for the different configurations is presented in Figure 4.19. In solid lines, baseline RTT and in dashed line, when the DRB is loaded with a 1 Mbps uplink UDP traffic.

The best way to reduce latency efficiently, is to assign all the bandwidth (all the PRBs of the next TTI) to the UE each time it requests radio resources (i.e. at the reception of a SR or a non-zero BSR), but this implies a low spectral efficiency, an important

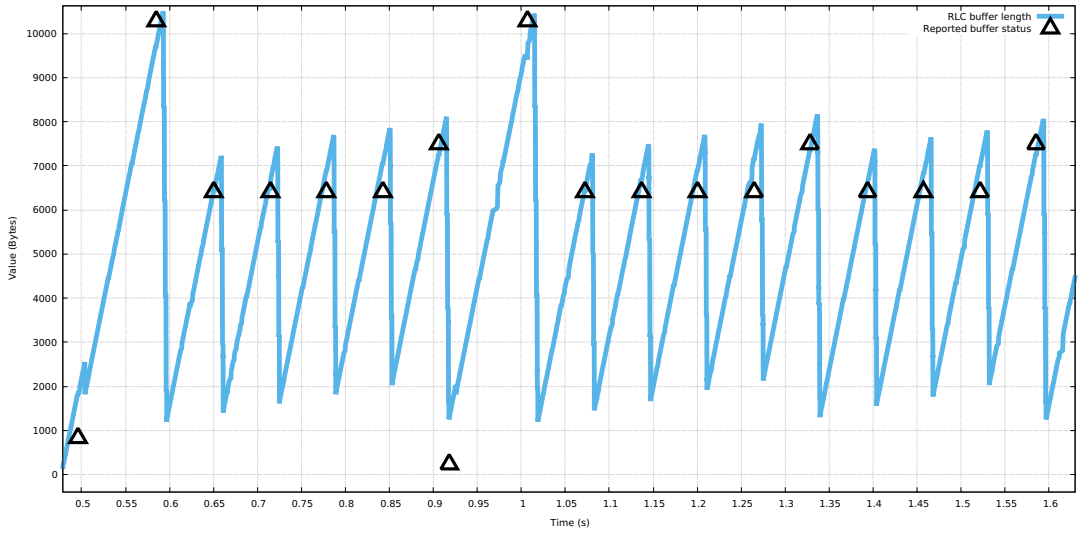



Figure 4.20 – Buffer reported (*black markers*) with BSR (in bytes) versus actual RLC transmission buffer occupancy (*blue line*).

transmission and coding energy costs for the UE and an unfair sharing of radio resources between UEs. Without modifying the scheduling algorithm, both reduction of the SR and BSR periodicity reduce the latency compared to the baseline configuration. The reduction of the SR periodicity to 1 ms clearly reduced the uplink latency and more importantly, it reduces the dispersion around the median when the ICMP flow is alone, whereas the reduction of the BSR periodicity to 8 ms reduces the latency but not the dispersion compared to baseline. When the ICMP flow shares the DRB with the UDP flow, the best strategy to reduce latency is to reduce the BSR periodicity (the median of the baseline is 55 ms, 50 with a short SR periodicity and 35 ms with a short BSR periodicity). Reducing the BSR periodicity has a beneficial consequence on the uplink latency jitter by reducing the size of the transmission bursts and the effect of "small transmission packets between big bursts".

In conclusion, to reduce baseline latency, we should select a SR configuration such as it increases its number of transmission opportunity but to reduce jitter and burstiness of a traffic flow, the buffer status reporting periodicity is the best trade-off configuration.

### 4.2.5 On the buffer knowledge

The allocation of the radio resource grid to devices requires knowledge of their buffer status for an efficient allocation of radio resources according to the needs. The detailed

analysis in the preceding paragraphs of the standards (§1.1.3.2), of the source code (§4.2.3) completed by experiments in Section 4.2.4 inform us that the base station's knowledge of the UE's buffer status are directly related to the transmission of the BSR. The illustration of that is plotted in Figure 4.20 where the ground-truth RLC transmission buffer occupancy is in blue and the reported value represented with black triangles (data from the UDP uplink experiment in paragraph 4.2). After the reception of each BSR, a corresponding UG causes the emptying the buffer of the corresponded reported value. It lets the buffer not completed emptied as there are a delay between the transmission of a BSR and the corresponding TB transmission. Moreover, the indexing of the reported byte volume induces an inaccuracy of 15% for 6000 bytes in the buffer. This inaccuracy in the reporting is clearly visible on the figure with 4 BSR reporting index 44 after second 0.6 and on $BSR$ reporting index 45 at second 0.9 by a threshold effect, changing the grant size of about 1000 bytes.

The figure is very illustrative of the buffer status sampling problem posed by the periodic buffer status reporting with a long period. The knowledge of the buffer state that the scheduler has is limited. The importance of the information reported by BSR for the scheduler is discussed in [66] in particular for delay-sensitive traffics.

In 2009 in his thesis [61], F.D. Calabrese dedicates the section §7.4.2 to the question of the impact of buffer knowledge on the streaming service. He shortly compared the throughput performances according to 2 types of traffic, a bulk CBR traffic with large packet and a bulk CBR traffic with small packet (768 Bytes) for an ideal knowledge of the buffer status and partial knowledge with a buffer status reports (the periodicity is not detailed by the author). He found that the partial knowledge impacts the average cell throughput, especially for the uplink small packet case with a loss of 8% and an increased bandwidth utilization due to padding and delays. In his work, he did not develop the latency and burstiness aspect related to the partial knowledge of the buffer status.

### 4.2.6 Related observations

Some work already noticed the intrinsic bursty nature of the uplink access [168][205]-§4.2.5. However, it is often hide and explained as the capacity variability of the air interface, the scheduling and the HARQ retransmissions [197]. We found 3 of them which notes the link between the latency and the jitter with the uplink traffic profile and the dynamic scheduling.

In *"End-to-end delay jitter in LTE Uplink"* (2020) [206], Sahu models and addresses

the problem of achieving a delay jitter target for uplink RAN. Especially, this work provide a model of uplink scheduling flow in NS3 and with the associated queue length process. He uses an M/G/1 fluid model to estimate the delay jitter. A strong limitation we found, he indicated himself is the manner of BSR is generated. As he wrote *"When an IP packet arrives in the PDCP queue, UE [as it does not have an uplink allocation] firstly sends an uplink SR to the eNB so that eNB can give an initial uplink allocation. This generally happens in LTE and not valid for NS3. In NS3, directly BSR is generated which indicates the amount of resource requirement and in the very next subframe, the eNB MAC scheduler gives the allocation of TBS (in bytes) for the corresponding BSR."*, but that is not true in both OAI, srsLTE and deployed networks. Thus, it models correctly the dynamic of queue length variation in time according to grant allocation, but it does not model the lack of knowledge on buffer status as we pointed out.

Bruhn, Kuehlewind and Muehleisen in "*Performance and Improvements of TCP Cubic in low-delay cellular networks*" (2022) [207] studied the link between delay variations in mobile networks and SR procedure (see II.A and II.C) in an Ericsson RAN testbed similar to the commercial network setup we used in §4.2.2. They depict in Fig. 3.a) the RTT sampling with ACKs over time and show the important variation of delay from one burst of transmission to another. Their analysis of the situation is in agreement with ours, but we think from our experiments that they underestimate the role of the BSR in the jitter and the creation of transmission bursts.

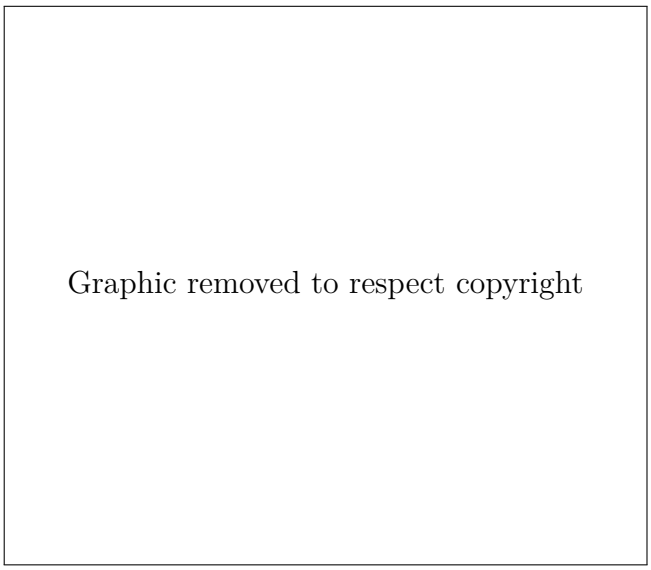



Figure 4.21 – Traffic profile of uplink ACK flow extracted from [203].

In a work concerning *low latency congestion control algorithm* for mobile traffic, Park et al. (2018) [203] pointed out in section §3.2 the difference between downlink and uplink traffic profile due to the grant-based uplink dynamic scheduling process. Figure 7 in this work illustrates well the difference of scheduling between downlink and uplink because of the SR periodicity. They correctly analysed the importance of the SR periodicity on the RTT variation over time. Especially in their work, they propose in section §4.4 a method to calculate the minimum RTT of the RAN from the uplink traffic observation and an estimation of SR periodicity. With the arguments we develop earlier in this chapter, we think they have missed a part of the analysis on their presented uplink traffic profile by not taking into account the contribution of the BSR to the transmission bursts which we see repeating every 40 ms or so (indicating a 32 ms BSR periodicity configuration in the given example).

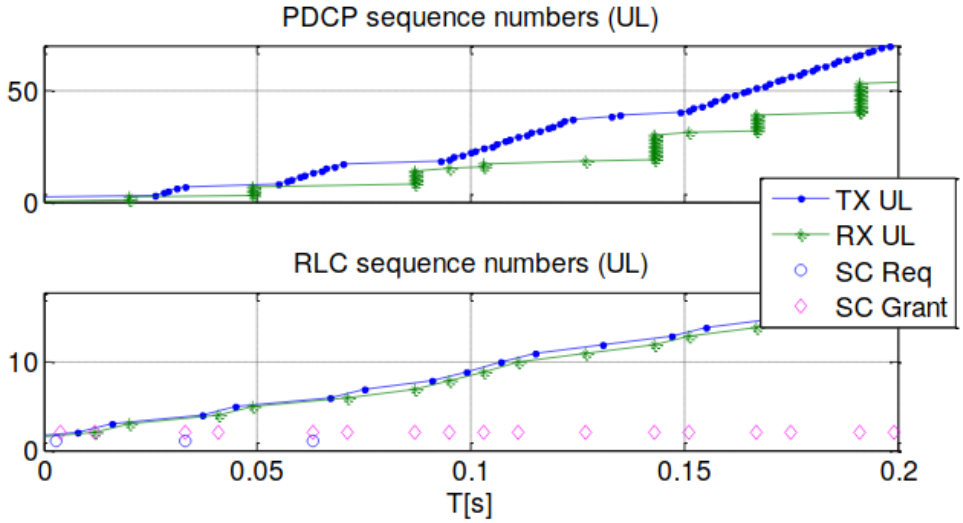



Figure 4.22 – Traffic profile of an uplink transmission reported by Johansson at the IETF 96 (2016) [208].

Ingemar Johansson from Ericsson already exhibited in [208] and [209] (2016) the bursty nature of the uplink radio access network. Especially he pointed out the fundamental difference between uplink and downlink scheduling where packets are transmitted as quickly as resource allocation allows with a "smooth" pattern when the uplink scheduling shows bundled packets that affect RTT. Figure 4.22 extracted from the IETF presentation of Johansson plots the uplink traffic profile at the PDCP layer at the UE (blue) and the base station (green) and at the RLC layer. Moreover, on RLC plot he indicates the SR and the UGs. The uplink traffic profile demonstrates that packets are waiting for an UG

in the RLC transmission buffer. However, we regret that the BSRs are not reported on the plot and are not studied as a source of transmission bursts. This work of Johansson is part of larger study on transport protocols and 4G/5G networks [209]. Thus, it focuses on the consequence of the bursts and RTT jitter more than on the analysis of the root causes.

## 4.3 Conclusion

In this chapter we investigated latency characteristics and jitter in the LTE RAN with an OpenAirInterface lab testbed.

The first section characterized the RTT of the RAN and the bufferbloat with a focus on the uplink channel. The uplink channel is less studied in the literature in spite of the interesting questions it poses. We dedicated the second section to the study of uplink traffic profile, and latency. We found a link between the access latency, latency variation and the grant-based allocation scheme. In particular this study is based on both controlled-variable experiments, commercial network experiments, analytical study of the code and on analysis of related observations from the literature.

The conclusion is that uplink dynamic allocation parameters made with a concern for a compromise between performance and control/energy costs can lead to important delays and above all significant jitter with bursty transmissions. The question, already summarized by Johansson in [209] that comes is the impact of the uplink channel traffic profile on the data flow and services.

Chapter 5

# Why Tackle the Uplink-Channel Latency and Jitter ?

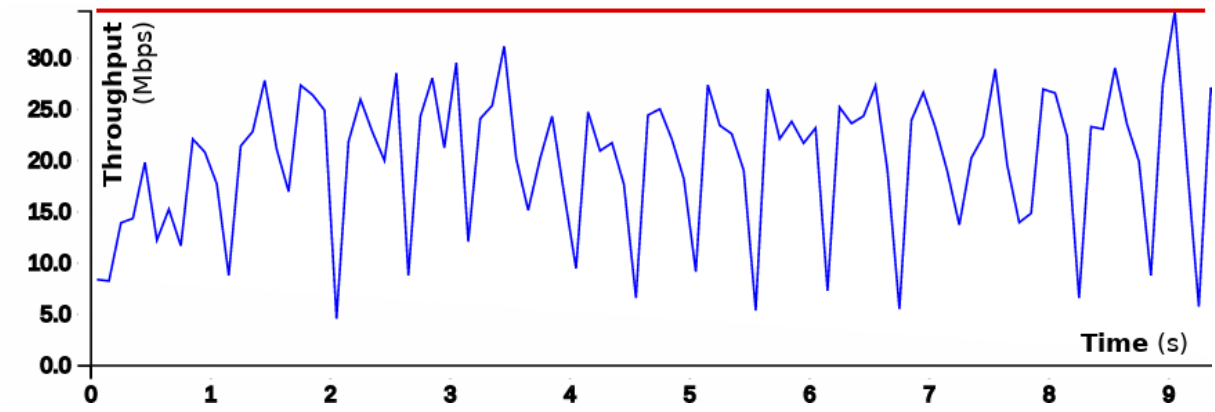



Figure 5.1 – TCP Cubic instantaneous throughput in blue compared to the available LTE radio link capacity in red.

**TCP inefficiency over cellular networks** Legacy congestion control protocols such as TCP and its variants are known to perform inefficiently over cellular networks. In particular, they fail to take advantage of the whole cellular network capacity. This observation is made by a certain number of studies [6, 205, 210, 211]. In [85], Liu et al. found a network capacity usage of 43.7%, 53.9% in [174], 52.6% in [212] and in [6], authors found an LTE network capacity utilization by TCP of **68%** (with tuned parameters [105]-§V-C.2).a)). This even worse in 5G with an estimated utilization of only 22% of the available capacities [6]. The inefficiency of TCP over cellular networks comes from a highly variable channel capacities over short time scales (path loss variation and obstruction associated to mobility) and unpredictable changes in RAN latency (e.g. handover, network load, HARQ loss) [16, 104, 213]. From that matter of fact, the topic of TCP over mobile network has becoming very active [77, 81] since the apparition of mobile internet services with 3G and LTE, but the utilization of mobile network specific TCP variant in

the network remains limited [78].

**TCP Cubic experiment in testbed** TCP Cubic is one of the most widely deployed TCP variants on the today internet [78]. Figure 5.1 depicts the throughput achieved over time of TCP Cubic on the OAI testbed (see Table 2.2 for network configurations) with a constant network capacity. In this experimentation, 2 radio errors have been recovered by the HARQ process, resulting in an error-free radio link. The average achieved goodput for a 10 seconds transfer is 22 Mbps, **59%** of the network capacity. The throughput regularly decreases as the sender congestion control believes to detect an indication of a congested network. TCP CCA uses acknowledgments to detect packet loss in the network and react to congestion [214] (Fig. 1.7). For a downlink transfer, an ACK flow is carried by the uplink channel.

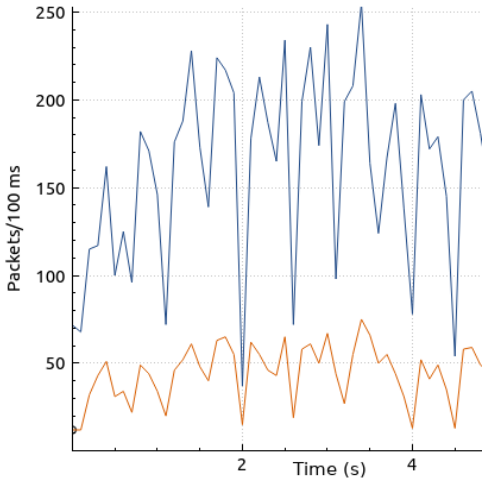



Figure 5.2 – ACK packet generation (orange) and Data packet generation (blue) per 100 ms over time

**Acknowledgments** Figure 5.2 plots the number of packets per 100 ms transmitted in the downlink and in the uplink. The Android TCP stack [1] generates in average 1 acknowledgment packet for 3.77 [2] data packets received (TCP ACK packet generation mechanism in [215]-§4.2 and [76]-§III). Thus, a dataflow of 22.6 Mbps in the downlink is associated to a 356 kbps uplink flow constituted of small packets of 80 bytes. An

1. Fork from Linux
2. ACK Ratio or AR

acknowledgment packet is generated every 2 ms, keeping the modem uplink radio interface always ON.

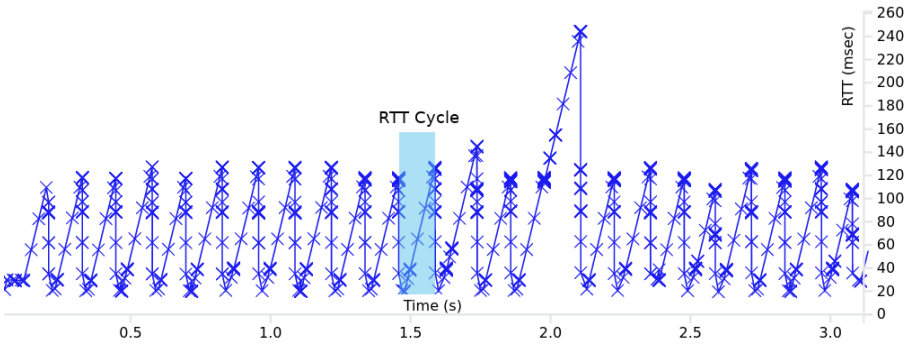



Figure 5.3 – Measured RTT by the TCP sender from ACKs over time.

**RTT and transmission pattern** In addition to loss reporting, ACKs are used by the TCP stack to estimate RTT of the network. Therefore, the sampling of the RAN RTT is directly correlated to the frequency of ACK generation. The RTT measured by the TCP sender over time is plotted in Figure 5.3 and shows important variation from one ACK packet to another. RTT varies from 30 ms to 120 ms with a mean at 60 ms and does not reflect the reality of the network status that is under-utilized. In this example, the RTT presents typical cycle between one extremum to another.

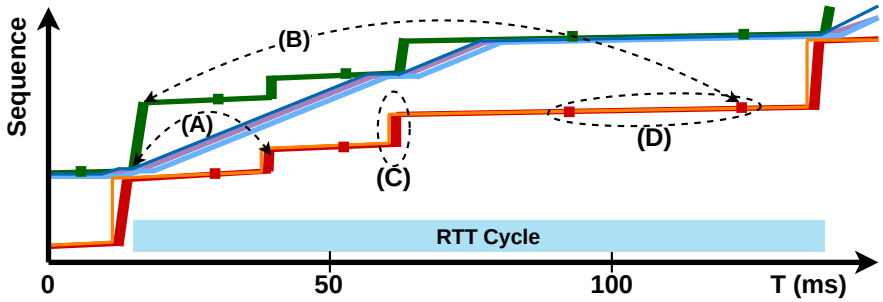



Figure 5.4 – Zoom on TCP traffic profile for a typical transmission cycle. *Green: At the entry of the BS (Downlink); Blue/Purple: at the UE (Downlink/Uplink); Orange/Red: At the output of the BS (Uplink).*

Figure 5.4 focuses on the traffic profiles corresponding to a RTT cycle. There are plotted the TCP traffic profile at the entry of the RAN in the downlink as it is transmitted by the TCP sender, the traffic profiles at the UE radio interface and traffic profiles at

BS's RLC layer in the uplink and at the output of the RAN. The RTT is measured between green and red curves. In (A), the RTT corresponds to the minimum RAN RTT as the downlink queue is empty and an uplink transmission opportunity is available. In (B), the last data packet of the initial TCP transmission burst is composed of 25% of queuing latency in the downlink and 75% of access latency due to a lack of UG; Burst of transmission in (C) after the reception of a BSR by the BS and packet transmitted alone in (D) even though the UE transmission buffer is not empty have been discussed in Section 4.2. It can be seen from figure 5.4 the under-utilization of the air interface capacity with the "pauses" in the transmission of the purple traffic profile. The purple line slope when the downlink buffer is non-empty corresponds to the peak capacity of the downlink interface, i.e. 37.5 Mbps.

**Congestion control** The lack of data in the downlink buffer is due to 2 TCP mechanisms related to the congestion window. The congestion window maintains the number of bytes in the network not yet acknowledged below the limit fixed by the congestion control algorithm. The reception of an ACK frees up space in the window, allowing TCP sender to transmit new bytes. That is why after the reception of a burst of ACK, the sending of a burst of new data in the downlink can be observed. A solution adopted by TCP BBR [216] is to pace [77] the sending datarate to the estimated network bottleneck capacity. Pacing keeps the downlink *buffer just full but not fuller*[3], preserving downlink latency from buffering delays. However, the packet sending rate is still limited by the congestion window.

The congestion window is controlled dynamically by the TCP CCA. The objective of CCA is to maintain and adapt a congestion window such as the sending rate is equal to the network bottleneck capacity (e.g. radio link capacity) so avoid buffer overflow losses. TCP Cubic CCA depends on packet loss (or estimated loss after DUP ACK and Retransmission TimeOut (RTO) expiration) as many other TCP variants (i.e. loss-based) to detect congestion or changes in network conditions. Others variants use RTT rises to detect congestion (i.e. delay-based) and others combine RTT and estimated bandwidth such as BBR (i.e. hybrid). All of these algorithms have in common to be dependent of the smoothed RTT [218] calculated from ACK feedback loop and exposed by the TCP stack.

3. *Internet congestion control using the power metric: Keep the pipe just full, but no fuller*, Leonard Kleinrock [217]

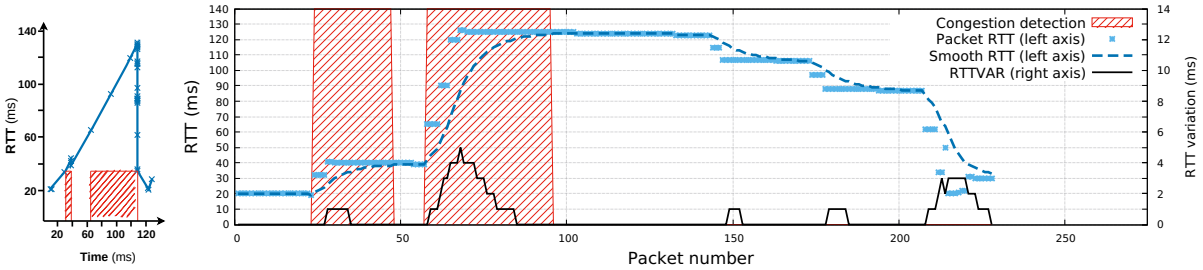



Figure 5.5 – Left : Packet RTT. Right : Packet RTT, smoothed RTT and RTT variation by the TCP stack for the typical transmission profile cycle in Fig. 5.4.

**TCP RTT measured from ACKs** Linux TCP stack [4] [219] computes the smoothed RTT using an exponential smoothing of parameter $\alpha = 1/8$ on a packet basis and not a time basis. The difference between the 2 basis is illustrated in Figure 5.5. On the left the packet RTT is plotted of time and on the right in blue, packet RTT plotted in the basis of packet sequence number. $RTTVAR$ corresponds to the variation of the RTT from one packet to another (black curve on Fig 5.5). The ACK bundling appears short on the time basis (2 ms) but long in the packet basis (150 packets). Hence, Smooth RTT variation on the right between packet 55 and packet 215 lasts for 2 ms. Combined, smoothed RTT and RTTVAR are used to detect congestion and calculate RTO [75, 220, 221] [5] [6]. Areas in red represents moments when $sRTT + 4 \times RTTVAR > RTT$, sign of a possible congestion [75]. According to this calculus, the network is most of the time in a congestion state even though it is in reality a lack of uplink grants to transmit ACKs. It is therefore not surprising to see Cubic struggling to increase congestion window.

**TCP BBR and Quic experiments** Under the same configurations, the state-of-the-art of CCA, TCP BBR [222] outperforms Cubic with 27.5 Mbps and **73%** utilization of the network capacity. However, BBR also suffers from the significant variation of RTT as it tries to reach optimal BDP with a varying delay [223].

The best network capacity utilization is obtained for *Quic* transport protocol (see an introduction to Quic in §1.2.3 and Quic traffic generation in §2.3) and BBR CCA with **86%** (= 32.2 Mbps).

4. Source code of the TCP RTT update function in Linux TCP stack: `tcp_ack_update_rtt(...)` `https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/net/ipv4/tcp_input.c?h=v4.15`
5. https://datatracker.ietf.org/doc/html/rfc6298
6. `http://sgros.blogspot.com/2012/02/calculating-tcp-rto.html`

In [83], authors evaluated Quic under different scenarios and also found Quic generally outperforms TCP, but they additionally identified performance issues on cellular networks with a poor radio resource utilization.

ACKs have been redesigned in Quic protocol [86] to carry more information [76] but less often [224, 225]. We got an Ack ratio from 1 : 10 to 1 : 36, depending on loss rate. As for TCP, Quic sending stack estimates from ACK sampling, the minimum RTT, the smoothed RTT and the RTT variation [86]-5. As there are 29 ms between each ACK, the uplink transmission of them is done similarly to the transmission of pings in section 4.1.1 with an uplink latency of 20 to 30 ms, less subject to RTT variation from one ACK to another. The reduction of the ACK frequency comes with a less reactivity to channel degradation, materialized by the increase of downlink packet delay associated to a buffer filling up.

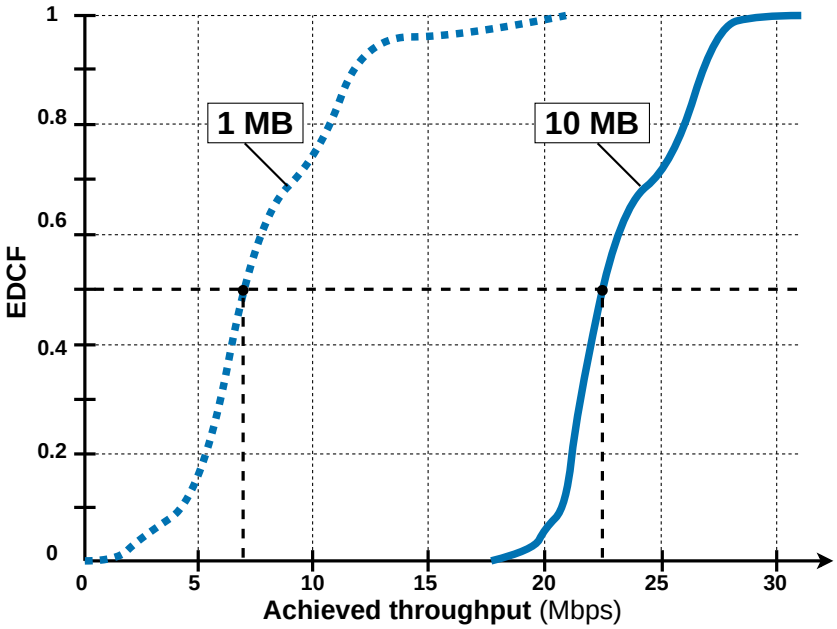



Figure 5.6 – Achieved throughput for a TCP Cubic (Hystart enabled) transfer of 1 MB and 10 MB.

**Short TCP transfer and Hystart** In addition to that, most applications are using TCP for short transmissions, such as Web browsing that opens multiple TCP connections to load objects from different servers [7]. At the start of the flow, TCP is continually

7. 72 objects per webpage in average according to live measurements of `https://webview.orange.com`

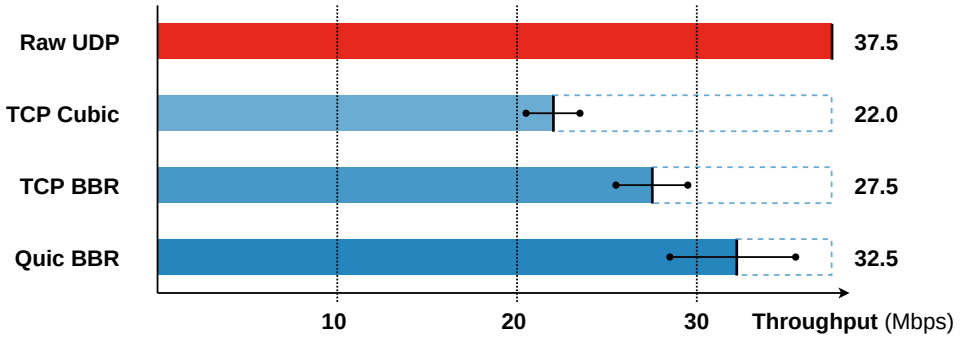



Figure 5.7 – RAN capacity utilization.

seeking out network bottleneck capacity by increasing its rate until it detects a congestion, following which TCP enters into congestion avoidance phase. Slow-start phase that characterizes TCP connection beginning has already been identified as a limitation [8] in LTE networks [212, 227–229]. Exiting the start-up phase too quickly [207] does not allow network capacity to be reached quickly at the start of the connection. Cubic CCA takes several rounds trip to reach the estimated network capacity after an early exit from slow-start. That is illustrated in Figure 5.6 with the ECDF of throughput achieved for TCP connection carrying either 1 MB or 10 MB. The mean achieved throughput of a 10 MB transfer is 22 Mbps while it is only 7 Mbps for a 1 MB file. Similar observations are made in [228, 230–232], impacting HTTP transfer duration. In average, TCP exits slow-start after 175 kB transmitted (i.e. 120 ms), and hits for the first time the network capacity after having transmitted 3.2 MB (i.e. 1.65 sec). The early exit of slow-start is triggered by the increase of latency, as an implicit sign of congestion (example in Fig. 4.5b), the same sign of congestion as in Fig. 5.5. Tackling latency jitter at the beginning of the TCP transmission has multiple positive impacts : It increases radio resource utilization in downlink; It makes the transfer duration shorter for a better user experience and also reduces the time during which user modem has to be turned on.

**Summary** The uplink latency and jitter is an issue for TCP transport protocol to efficiently use radio access network capacity (figure 5.7). By delaying ACKs, the uplink channel hides the congestion state of the downlink channel, which fools the transmitting TCP stack and leads to falsely congestion detection. That is particularly visible at the beginning of the connection where TCP seeks to reach network capacity but fails because of an increase of RTT due to uplink allocation scheme. Moreover, the inter-arrival times

8. Especially Hystart algorithm [226] that is by-default used by Linux TCP stack

and bursts sizes fluctuate such in it does not follow a simple Poisson process as assumed in [233] and in several mobile network queueing models.

The next generation Quic protocol is also sensible to these variations of RTT to estimate the network congestion status. The issue should be the same and even worse in 5G with many latency challenges associated to it: highly variable radio channel capacity [9] [102, 211, 234, 235], frequent handover, buffer sizing and blockage effect [82][81]-Table 2.

**Communication**

— *Considering return path latency in access networks.* **Flavien Ronteix–Jacquet**. NetSatDay meets NoF: "Fast convergence of congestion control", July 2021, Toulouse / Virtual, France. Link to presentation : `https://hal.archives-ouvertes.fr/hal-03327308`

9. Figures of throughput in [102] are very illustrative of variability of the 5G mmWave channel

Chapter 6

# Enhanced-BSR

**Abstract** : *In LTE and 5G mobile networks, radio resources are dynamically allocated. In the uplink, the Base Station (BS) needs to know how much data are awaiting in the transmission buffer to be transmitted for each device to properly share the radio resources among them. In this chapter, using a simple scenario and measurements made on an operational network, we show that the BS works on systematically outdated information about the buffer size, which generates delay spikes. Thus, we propose to add a predictive estimation of the buffer size, called enhanced Buffer Status Report (eBSR). It is implemented in OpenAirInterface BS at the Medium Access Control (MAC) layer without changes in the scheduler, and it is compliant with LTE and 5G. We evaluated it on a lab testbed with various traffic profiles. Results show a clear improvement on uplink latency and jitter, together with a limited impact on radio resource usage efficiency.*

## Contents

**6.1 Needs of an Improved Buffer Estimation Method . . . . . 139**
6.1.1 Context . . . . . 139
6.1.2 Our approach . . . . . 140
6.1.3 Related work . . . . . 140
**6.2 Design and Implementation . . . . . 142**
6.2.1 Estimation algorithm principles . . . . . 142
6.2.2 Prediction for the uplink dynamic scheduler . . . . . 145
**6.3 Evaluation . . . . . 146**
6.3.1 Lab testbed . . . . . 146
6.3.2 Results . . . . . 147
**6.4 Conclusion and Further studies . . . . . 153**

**Publications**

— *Rethinking Buffer Status Estimation to Improve Radio Resource Utilization.* **Flavien Ronteix–Jacquet**, Xavier Lagrange, Alexandre Ferrieux, Isabelle Hamchaoui. IEEE VTC'22 Spring, June 2022, Helsinki, Finland. DOI : `https://doi.org/10.1109/VTC2022-Spring54318.2022.9860632`

— *Optimisation de la gestion d'allocation ressources radio à travers un réseau d'accès cellulaire.* **Flavien Ronteix–Jacquet**, Alexandre Ferrieux, Patent nb. 20 2424. 18/10/2021 (FR 2111020).

# 6.1 Needs of an Improved Buffer Estimation Method

## 6.1.1 Context

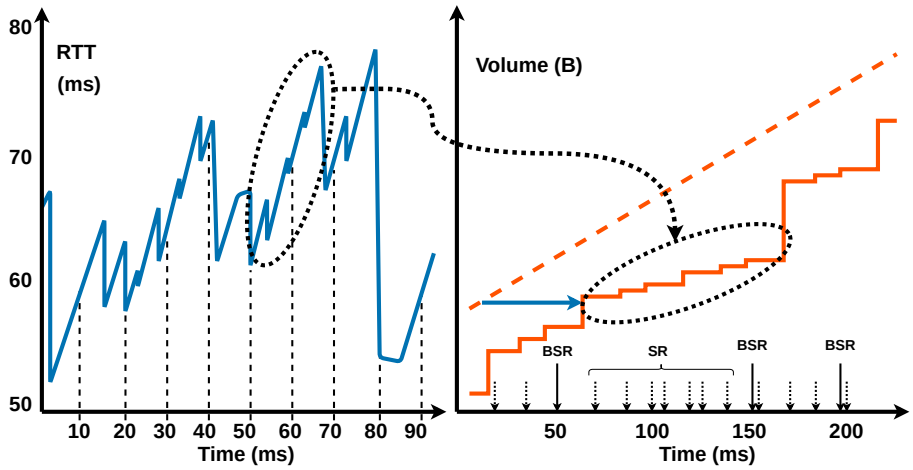



Figure 6.1 – RTT and uplink traffic profile of a TCP downlink transmission in a commercial network (see §4.2.2 and [4]); Left: RTT variation over time; Right: Uplink traffic profile. *Dashed orange line for uplink traffic profile before the air interface; After the air interface at the BS with the solid line. Guessed SRs and BSRs signalling are indicating on the bottom of the plot.*

In section 4.2 of Chapter 4 we exhibited with an uplink transfer how the current grant-based access method generates latency and bursts. Figure 6.1 illustrates the link between the RTT we observed on a commercial network (see setup in Sec. 4.2.2 and publication [4]) and the traffic profile captured showing transmission "steps". The circled region shows an increase of the latency due to a lack of grants. The corresponding uplink traffic in exhibits a very bursty profile, in line with related observations presented in section 4.2.6. Bursts of transmissions (big step on the traffic profile) occur after a BSR is received. Such delays and delay variation are not justified by a radio capacity bottleneck or by retransmissions due to poor radio environment, but only by allocation based on grants and BSR. BE default radio bearer utilizes the grant-based scheme for uplink radio resource allocation.

In the same time, we motivated the needs to minimize bursty uplink transmission pattern and jitter in Chapter 5 with the example of TCP. TCP congestion control has difficulties to adapt its sending transmission window to the network capacity variability and quickly varying RTT. Thus, the network capacity is not fully utilized (see Fig. 5.1) that is a source of internet latency identified in paragraphs 1.3.3 and 1.3.2. TCP represents

the major part of the internet traffic, that is carrying by the default BE bearer in the RAN.

### 6.1.2 Our approach

Our approach redefines the estimation of UEs buffer status at the BS left unmodified the grant-based access method, the BS's MAC scheduler and the UE unchanged. A continuous estimation of the UE's transmission buffer is performed at the difference of the sampling made with "legacy" BSR mechanism. The model of estimation for each UE uses BSR measurements and the transport block utilization rate and returns the estimation buffer occupancy. We successfully show that a simple linear estimation model reduces effectively uplink latency and jitter for non-QoS flows. More than an algorithm, we propose a new architecture for the uplink scheduler that includes an estimation model.

### 6.1.3 Related work

The firsts relative work concerning estimation of transmission buffer length is the implementation done in RANs using BSR. We studied it in Section 4.2.3. The conclusion is that estimation is partial and depends on a BSR reception event.

Several other studies have proposed model of estimation for the purpose of radio resource allocation. Applied to SPS or Fast-UL (§1.1.3.1), the estimation is used to pre-schedule grants before the UE has to request it with a SR or a BSR. Fast-UL especially is an efficient solution to lowering uplink latency [208]. [236] uses a predictive pre-allocation for low-latency uplink access in the industrial wireless networks. [237] prohibits unnecessary SR for uplink grants by using the SPS for lower priority data to not delay high priority which uses Sounding Reference Signal (SRS). [238] is a predictive scheduling to solve delay problem through proactively scheduling a mobile station without actual knowledge of the data it wants to transfer. A close work to our proposition is the prediction based on transmission history proposed by [239] for SPS. They show how the prediction of size new arrived data to adapt semi-persistent scheduling is beneficial for uplink latency. Their approach needs to be implemented in both UE and BS because of SPS configuration and does not take advantages of dynamic scheduler strategy. The prediction method is also different as our approach, based on auto-regressive model using the buffer occupancy time series. The patent [240] proposes to predict using the history of packet generation the arrival of new packet to pre-allocate radio resources. These works are designed towards

specific services such as industry machine to machine communications. For a majority of them, they are taking place in grant-free allocation schemes with SPS and does not take advantage of BSR reporting, dynamic MAC scheduling and modem optimizations. However, mathematical models for estimation developed in these work have to be taken into account when designing an traffic generation estimation algorithm.

In §4.2.4, we concluded the decrease of BSR transmission periodicity diminishes the uplink latency of long transmissions. Instead of reducing the periodic BSR timer value, other studies propose to better estimate the buffer occupancy reported to the BS, as proposed by [241], taking into account the so-called *buffer reporting delay*. In that work, the BSR value is modified in the UE, to integrate the estimated data filled in the buffer during the time for the BSR to become effective. Unfortunately, this attractive approach requires a modification of the UE, which may reveal quite difficult to implement and deploy. In a recent work [242] (2021) Zhang also tries to predict the BSR to integrate information of data arriving after the transmission of the BSR for reducing uplink scheduling latency. For that, he evaluates several state-of-the-art supervised machine learning algorithm such as random forest, XGBoost and Long Short Term Memory (LSTM). However, this solution does not solve the problem of the continuous estimation of the UE transmission needs. He solely proposes to palliate a limitation of the BSR, which is the delay between its reception and the effective transmission following an UG. [66] proposes a "*No Delay Information*" scheme to limit the feedback buffer status reporting schemes for delay-constrained DRB. They illustrate by their solution that delay information at the base station is important for scheduling real-time flows. [151][243] propose to estimate delay experienced by packets from BSR (not the actual queue length) for delay-aware scheduling purpose. In [244], the queue length estimation is also used for a better congestion control in bandwidth-varying mobile network. The queue length estimation is also used like us for the uplink dynamic scheduling [245][246] but their approach of estimation is different and less general. Guo et al. in [22] also highlights some limitations of BSR in §6.2 and proposes, as a part of a bigger solution, a buffer estimation method in order to reduce on-device delay (i.e. uplink bufferbloat). The authors combine BSR, padding statistics (i.e. transport block utilization) and grants to estimate throughput and finally derives from it an estimation of transmission buffer length. This solution is implemented at UE's modem for active queueing management purpose, so the estimated value cannot be used by BS's scheduler for resources allocation.

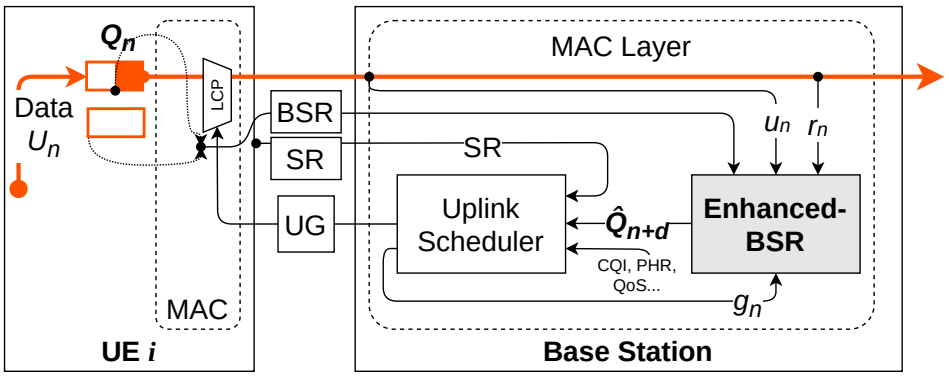



Figure 6.2 – Proposed uplink scheduling with enhanced-BSR algorithm.

## 6.2 Design and Implementation

We propose to include a new module called *Enhanced-BSR* (eBSR) in the MAC entity of the BS as shown in Fig. 6.2. Its role is to provide an estimation of the UE buffer occupancy that is more frequently updated than the BSR. The estimation is based on uplink signalling i.e. BSR, SR, padding size. The proposed estimation algorithm models the traffic source as being of constant bitrate over a time period.

### 6.2.1 Estimation algorithm principles

#### 6.2.1.1 Queuing model

In our proposal each UE $i$ is modeled as a queueing system that is filled by a single traffic source of piecewise constant datarate $D$ and emptied by uplink transmissions $L$ following a UG $g$. At slot $t$, the number of bytes $Q_t^i$ in the buffer of UE $i$ is given by:

$$Q_t^i = [Q_{t-1}^i + D_t^i \cdot T_{\text{tti}} - L_t^i]^+ \tag{6.1}$$

where $D_t^i$ is the packet generation datarate, $T_{\text{tti}}$ is the slot duration (e.g. $T_{\text{tti}} = 1$ ms) and $L_t^i$ is the number of bytes correctly received by the BS. All the above parameters are known to the UE. The BS knows $L_t^i$ but neither $Q_{t-1}^i$ nor $D_t^i$. The computations below are per UE but for the sake of simplicity, we remove the upper index $i$ in the following.

The objective of the estimation model is to provide the BS with an estimate $\hat{Q}_t$ of $Q_t$. At each time slot and for each UE, the base station computes:

$$\hat{Q}_t = [\hat{Q}_{t-1} + \hat{D}_t \cdot T_{\text{tti}} - L_t]^+ \tag{6.2}$$

The key parameter in (Equation 6.2) is the estimated uplink source rate $\hat{D}_t$. To explain how $\hat{D}_t$ is computed, we introduce $d_t$ and $g_t$ as the number of bytes transmitted and the number of bytes granted for slot $t$, respectively. Note that $L_t \leq d_t \leq g_t$. When $d_t = L_t$, the transport block is correctly received by the BS (no HARQ retransmission). When $d_t = g_t$, all the resource blocks allocated by the BS are used by the UE. Even if the BS does not receive the transport block, it can determine $L_n$ and it knows $d_n$ and $g_n$. We define the transport block utilization computed at each reception of a transport block as $\rho_n = d_n/g_n$, note that $0 \leq \rho_n \leq 1$.

#### 6.2.1.2 Estimation update

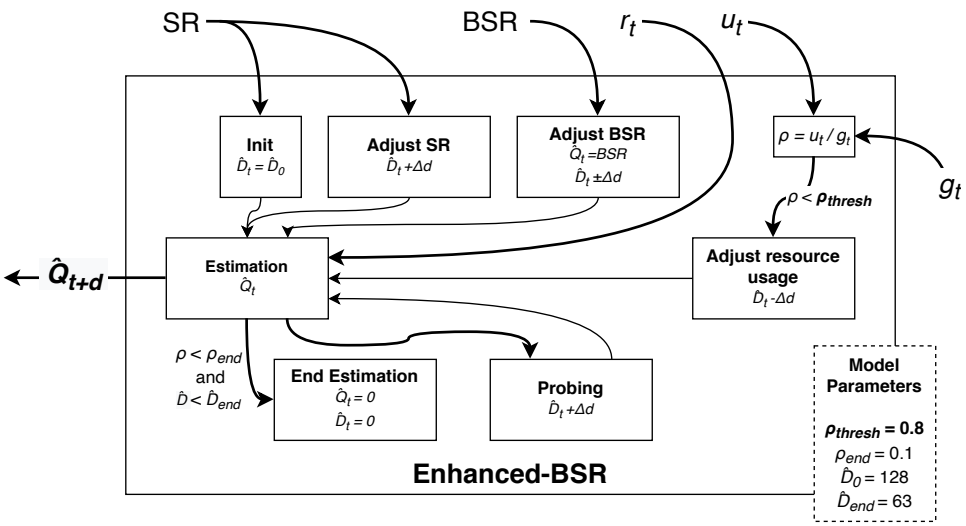



Figure 6.3 – Enhanced-BSR estimation model state machine.

Value of $\hat{D}$ is updated according to events as illustrated in figure 6.3. At reception of the first SR, variable $\hat{D}$ is initialized to a default value $\hat{D}_0$ (e.g. to 128 kbps, which is the LTE air interface capacity for a low channel quality) and then increased or decreased as described below.

After that, the update of $\hat{Q}$ is made continuously at each TTI with the equation 6.2 and $\hat{D}$ is updated at a reception of a SR, a BSR, of a TB or at the activation of a probing phase. $\hat{D}$ is increased and decreased by a value $\Delta d$ at the triggering of one event (*Adjust BSR*, *Adjust SR*, *Adjust PRB usage*, *Probing*). The estimation ends when the estimated traffic generation has stopped.

**Increase** $\hat{D}$ : A *non-zero BSR* reception indicates a data accumulation at the UE queue. $\hat{Q}$ is updated with the reported buffer size $Q^{\text{bsr}}$ and $\hat{D}$ is increased accordingly to account for the extra bytes generated in the last BSR period of duration $T_{\text{bsr}}$ according to:

$$\hat{D}_n = \hat{D}_{t-1} + (\sum_{m=n-T_{\text{bsr}}}^{n}(g_m - L_m) - Q^{\text{bsr}})/T_{\text{bsr}} \tag{6.3}$$

In Equation 6.3 the sum includes the difference between granted and estimated bytes in the last BSR period since the queue build-up could also be the result of the scheduling policy. We also propose a *probing* mechanism which aims to deliberately increase $\hat{D}$ for a period of time when no BSR has been received for a period of time.

Probing has been added because network-configured BSR periodicity could be as long as 1 second which incurs a delay to adapt to an increase of $D$. Probing phase tries to "sense" an increase of $D$ before the reception of a BSR occurs by a voluntary increase of $\hat{D}$. If a low $\rho$ is observed, the estimation undo to the previous $\hat{D}$.

**Decrease** $\hat{D}$ : A novelty of this work is the use of $\rho_t$ defined as the proportion of resource used to transmit user data. Its role is to monitor the effective use of the granted radio resource and the relevance of the estimation of UE transmission needs.

An important goal of the estimation algorithm is to avoid radio resource waste caused by an over-estimation of the UE buffer. A too large UG compared to the current buffer size is denoted by a low value of $\rho_t$. Thus, as soon as $\rho_t$ indicates wasted radio resources, $\hat{D}$ is decreased to lower the estimation of UE transmission needs. $\rho_t < \rho_{thresh}$ (with $\rho_{thresh} = 0.8$) triggers a decrease of $\hat{D}$ such that:

$$\hat{D}_n = \hat{D}_{n-1} \cdot \rho_n \tag{6.4}$$

The threshold of $\rho_{thresh}$ is a parameter of the algorithm. We use this value because it is common to set control plane overhead at 14% for goodput calculation [41] and because we observed $\rho_t > \rho_{thresh}$ when UG are correctly sized. Also, $\rho_t$ is not computed when a HARQ error occurs in slot $t$ as the content of the transport block in terms of user data, control signalling and padding is not known. When the traffic generation has stopped, $\rho_t$ is expected to be 0. We use this property to end the estimation when $\rho_t < \rho_{end}$ (with $\rho_{end} = 0.1$) and $\hat{D}_{t-1} < \hat{D}_{end}$ (e.g. $\hat{D}_{end} = 128$ kbps).

In practice, an **exponential smoothing** on $\hat{D}$ of parameter $\alpha = 0.8$ is used to converge towards $D$ while quickly reacting to $D$ decrease (An example of $\hat{D}$ convergence

is given in Appendix E.E.2a).

**Fallback to "legacy" BSR** : We employ a conservative method to preserve radio resource utilization which stops the estimation and fallbacks to "legacy" BSR estimation when the algorithm does not converge i.e. when the achieved $\rho < 0.8$ for 10 consecutive transmission in a time frame of 200 ms. The time frame is chosen to let multiple BSR transmissions and adaption rounds. With a BSR timer at 64 ms, it means the model could adapt estimated datarate 3 times from BSR. If during this phase of adaption, the model does not converge with a $\rho < 0.8$, then it means the source datarate is not deterministic enough. That is the case when the traffic generation pattern does not comply with the linear model of the proposed estimation.

### 6.2.2 Prediction for the uplink dynamic scheduler

The output of the algorithm is the estimation of the UE transmission buffer length $\hat{Q}_t$. This value is then taken into account by the uplink dynamic scheduler to determine UG size according to its own algorithm as illustrated in Fig. 6.2. The first contribution of our approach is to provide a continuous estimation of the UE buffer length from the Equation 6.2. A second important contribution is the prediction of the UE buffer state at the horizon of $d$ slots under the assumption of a relatively constant datarate of $D_t \sim D_{t+d}$:

$$\hat{Q}_{t+d} = [\hat{Q}_t + d \cdot T_{\text{tti}} \cdot \hat{D}_t - \sum_{m=t+1}^{t+d-1} g_m]^+ \tag{6.5}$$

Setting $d = 8$ for LTE is a wise choice since there are 8 subframes between the scheduling decision and the effective transmission by the UE. Thus, the uplink scheduler will send grants for slot $t+8$ not based on current $\hat{Q}_t$ but on expected buffer status $\hat{Q}_{t+8}$, effectively reducing uplink access latency for packets arriving in buffer at slot $t+7$.

## 6.3 Evaluation

### 6.3.1 Lab testbed

**Base Station** We evaluated the performances of the proposed *enhanced-BSR* algorithm on the lab testbed presented in section 2.2 with the configurations in Table 2.2. We have implemented our proposition in the OAI LTE BS as a MAC-layer module as presented in Fig. 6.2 (around 1.750 source lines of code). The MAC scheduler is left unmodified and selects UE to schedule as described in the Sec. 4.2.3 when an explicit SR signal is received.

**Air interface** In the following, the air interface is ideal (SNR> 12 dB) and does not vary in time. This strong hypothesis aims at assessing the intrinsic ability of our algorithm to correctly estimate UE transmission needs. The evaluation in more realistic conditions on an error-prone channel is left for further work, though anecdotal evidence on production networks tends to confirm the existence of this delayed estimation problem in real life.

**Traffic considered** We consider 5 types of traffic to evaluate different properties of our solution. Each traffic are generated independently.

ICMP *Ping* traffic is used to evaluate baseline RAN RTT. One 64 bytes-length ping packet is generated every 50 ms.

An *uplink traffic pattern* is generated using UDP packets to evaluate the scheduling performance in terms of latency and jitter. The latency is measured for each packet thanks to the simultaneous captures at the UE and the BS side. Two traffic patterns are considered. A 1 Mbps CBR traffic with a deterministic inter-packet delay of 0.8 ms (100 bytes-length packets), thereby the transmission buffer is filled every TTI. A non-deterministic traffic where inter-packet delay is variable and random (delay applied with tc netem tool [1]). The average inter-packet delay is 0.8 ms (100 bytes-length packets) with an average datarate of 1 Mbps. The traffic is then very hard to predict for a linear model if the instantaneous datarate differs too much from the average.

*TCP* BBR 30 MB-length transfer in downlink and uplink to show the impact of the uplink access method on throughput, delay and jitter experienced by users. It should be noted, the downlink TCP transfer generates an uplink flow of ACK with similar characteristics to an almost-CBR traffic when the TCP estimated bandwidth has converged.

1. Netem wiki : `https://wiki.linuxfoundation.org/networking/netem`

| Experiment | Metric | **BSR** = 64 ms | **eBSR** |
|---|---|---|---|
| Ping | achieved $\rho$ (%) | 56.2 | 31.6 |
| | RTT (ms) | | |
| | P10 / P50 / P90 | 25 / 31 / 38 | 13.5 / 20 / 24 |
| Uplink CBR | achieved $\rho$ (%) | 82.5 | 72.0 |
| | median RTT / Jitter (ms) | 52 / 47 | 19 / 10 |
| Uplink bursty | achieved $\rho$ (%) | 77.5 | 70.5 |
| | median RTT / Jitter (ms) | 55 / 50 | 42 / 35 |
| TCP Uplink | achieved $\rho$ (%) | 97.2 | 97 |
| | Throughput (Mbps) | 16.3 | 16.7 |
| | Latency (ms) | | |
| | P10 / P50 / P90 | 18 / 130 / 440 | 14 / 110 / 380 |
| TCP Downlink | achieved $\rho$ (%) | 97.1 | 86.5 |
| | Throughput (Mbps) | 28.0 | 32.0 |
| | Latency (ms) | | |
| | P10 / P50 / P90 | 25 / 55 / 125 | 28 / 32 / 60 |
| | Avg. DL RLC buffer (kB) | 293.2 | 186.6 |

Table 6.1 – Summarized enhanced-BSR evaluation results. *P10: 10th percentile; P50: Median; P90: 90th percentile.*

### 6.3.2 Results

Results discussed in this section are summarized in Table 6.1. Achieved $\rho$ corresponds to the overall transport block utilization achieved over all uplink transmissions. It is calculated by taking the total volume scheduled $g$ by the MAC scheduler and the total volume received $d$ in the RLC buffer. The control signalling at the MAC layer and Signal Radio Bearer (SRB) are then not included in the value $d$. The targeted achieved $\rho$ is 80% as justified in 6.2.1.2-"Decrease $\hat{D}$".

Regarding the impact of Enhanced-BSR on the MAC layer performance, it is limited by the fact that even if one model is instantiated per connected terminal, updating the model at each TTI corresponds only to a few linear mathematical operations. The main cost comes from the search for the UG corresponding to the reception of a TB, an operation which is already partly performed by the MAC layer. The prediction that is generated at each round of scheduling is also a linear mathematical operation that is quick to perform. Finally, the memory needed for each Enhanced-BSR instance is limited to few variables to maintain.

#### 6.3.2.1 Uplink Latency

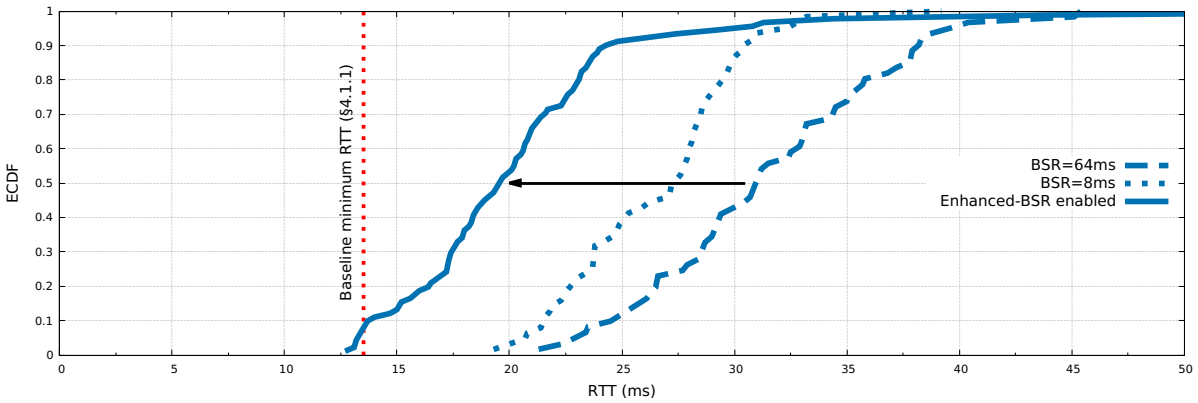


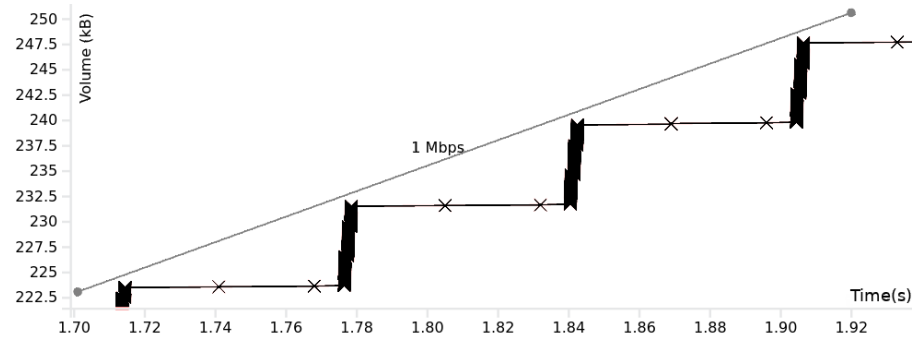

Figure 6.4 – ECDF RTT for BSR= 64 ms, BSR=8$ms$ and Enhanced-BSR enabled.

Figure 6.4 depicts the ECDF of a ping in optimal condition (no competition, no retransmissions, no variable channel capacity) for eBSR, BSR with a timer of 64 ms and one of 8 ms. The experiments show a clear improvement in terms of latency for 95% of the packet. The median RTT is reduced from 31 ms to 20 ms with a reduced jitter. Moreover, 10% of the packets experienced a RTT of less than 13.5 ms, the baseline minimum RTT (§4.1.1).

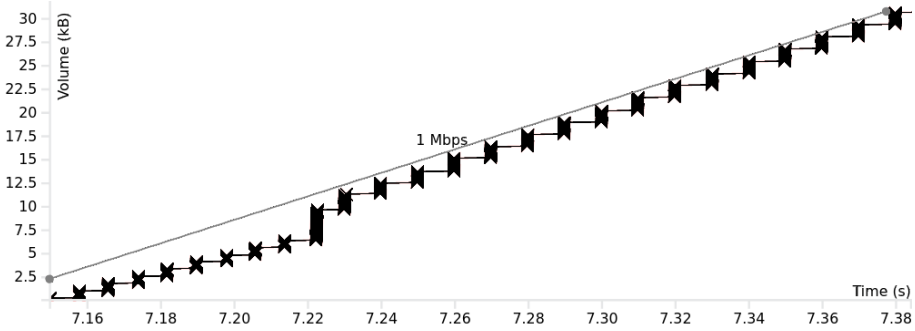
At the reception of a SR, the model is initialized with an initial estimated datarate. Because of that, the estimated buffer length is not zero and the scheduler will allocate radio resources as if it receives a BSR with the SR. The counterpart of this is a poor estimation of the buffer length, leading to a $\rho$ of only 31.6%. After the reception of a padding TB, the estimation is stopped. There are no obvious solution to set model initial values since it depends on the type of traffic that is not priorly known. The model converges with the firsts value of transport block utilization and BSR.

#### 6.3.2.2 UDP uplink transfer with eBSR

**Constant BitRate traffic** That is the best case because this traffic pattern matches the modeling of the source datarate as being of constant bitrate. The difference between the baseline and with our proposition is made clear with the uplink traffic profiles presented in Figure 6.5. At the top (Fig. 6.5a), we found the characteristic traffic pattern studied in Sec. 4.2.1 with bursts of transmission after the reception of a BSR. At the bottom (Fig. 6.5b) the traffic pattern is "stepped" with a pace of 10 ms, corresponding to SR




(a) With BSR= 64 ms




(b) With Enhanced-BSR enabled

Figure 6.5 – Enhanced-BSR results on uplink traffic profile for a 1 Mbps CBR transmission.

triggering periodicity.

According to the results presented in Table 6.1, the latency is reduced (visible also on the traffic pattern) but more importantly, the jitter (difference between 10th and 90th percentile) is crushed from 50 to 10 ms thanks to the more regular transmission of packets with burst of same size in opposition to the baseline transmission. It complies with latency/jitter-sensitive applications needs with a limited impact of the transport block usage.

Moreover, the convergence of the model is visible on the traffic profile 6.5b where $\hat{D}$ converges towards $D$ after few scheduling rounds. Before second 7.22, the traffic profile slope is lower than 1 Mbps which means an under-estimation of the model, corrected after the second 7.22 and the reception of a BSR. After that, the estimated datarate $\hat{D}$ is equal to the actual source datarate $D = 1$ Mbps with a good transport block usage around 80%, the model objective (see Appendix E.E.2a and E.E.2b respectively).

**Bursty traffic** That is the worst case for the prediction because of the non-deterministic nature of this traffic in contrast with the deterministic nature of the modeled source datarate. The latency and jitter are not as reduced as for the deterministic traffic with a poor transport block usage ($<<$ 80 %) resulting in a fallback to "legacy" BSR estimation (see paragraph §6.2.1.2-"Fallback to legacy BSR"). The fallback to the legacy is a conservative policy to preserve TB utilization in sacrificing latency jitter.

#### 6.3.2.3 TCP transfer and eBSR

We motivated in Chapter 5 the need to tackle latency and jitter of the uplink channel by TCP transmissions. Contrary to the use case presented in Chap. 5, we prefer to evaluate TCP performances with BBR CCA instead of Cubic because we think BBR will be adopted widely in the upcoming years. Also, even if BBR is the state-of-the-art of TCP CCA in terms of throughput (see Fig. 5.7), it does not perform optimally over cellular networks [247], that constitutes an interesting challenge for our proposition and an argument for its adoption in cellular networks.

**Uplink transfer** In the uplink, we do not observe a clear improvement either on latency or on throughput. In fact, TCP BBR has a good utilization of the uplink radio capacity of more than 95% of the available capacity. Our proposition has less impact on the uplink traffic pattern than the scheduler in itself when TCP reaches the radio capacity. The internet TCP congestion control machine maintains a number of bytes in the network contrary to the UDP traffic seen before. Thus, even when scheduler over-estimates the UE transmission needs, the TBS is filled with TCP packets present in the transmission buffer. It explains the almost full usage of the TBS measured with the achieved $\rho$.

Also, our solution improves the first megabyte transmit duration on average in 450 ms instead of 570 ms thanks to a delayed exit of the slow-start phase.

**Downlink transfer** The TCP downlink transfer experiment in the proposed setup is characterized by the presence of 2 feedback loops of estimation. The one of the server TCP stack and the one of the enhanced-BSR model as illustrated in Figure 6.6. The TCP estimation loop probes the RAN capacity using the ACK flow. Uplink channel latency and jitter disturb the ACK flow impacting the TCP bandwidth estimation. The enhanced-BSR model estimates the datarate of ACK flow using the TB usage and BSRs. When the TCP throughput raises up, the number of ACK increases that is detected by

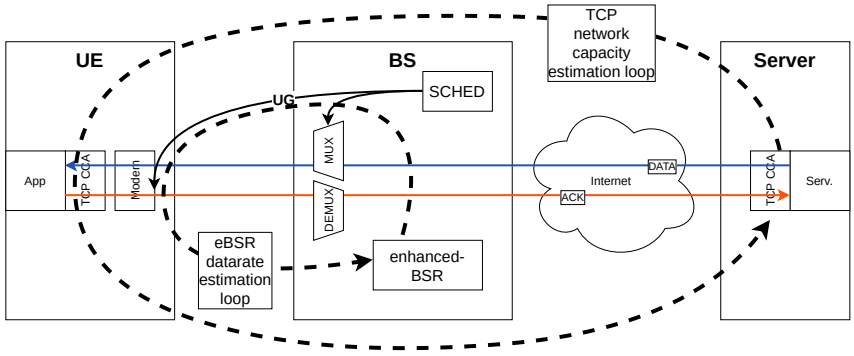
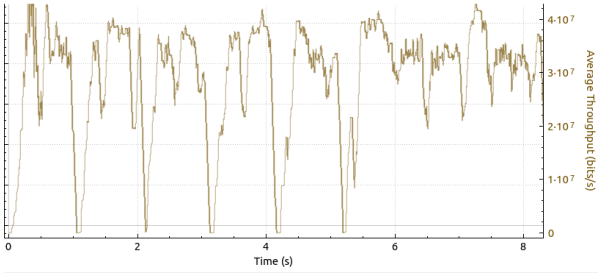
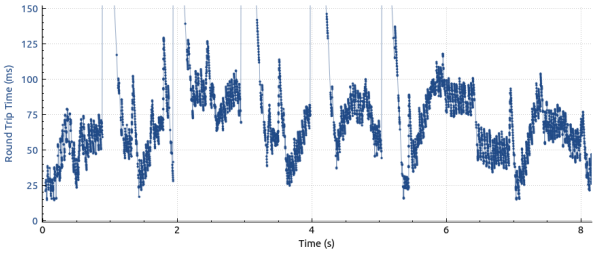



Figure 6.6 – Double feedback loop with TCP and enhanced-BSR.

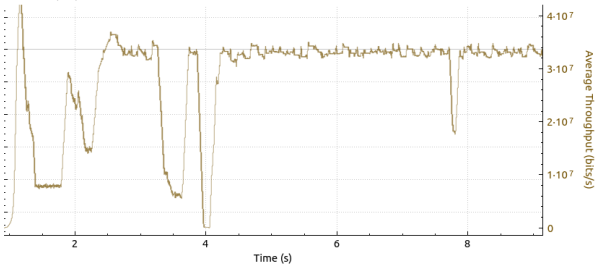
the model to adapt estimation revised upwards. A consequence of that, is an achieved $\rho > 80\%$ even if the model tries to converge towards 80%.



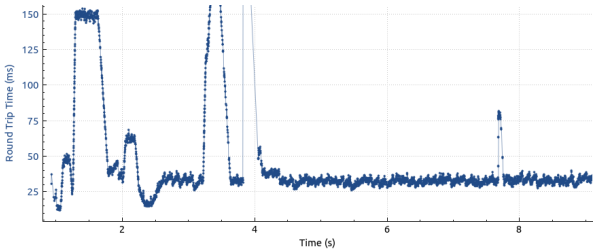

(a) Average throughput with BSR = 64 ms




(b) TCP RTT with BSR = 64 ms




(c) Average throughput with Enhanced-BSR




(d) TCP RTT with Enhanced-BSR enabled

Figure 6.7 – TCP BBR downlink transfer results.

In the downlink, we observe with BBR a minor average throughput improvement from 28 Mbps to 32 Mbps but with a greater stability of the instantaneous throughput as depicted in figures 6.7a and 6.7c. Especially, with enhanced-BSR TCP BBR converges to the estimated network capacity after 3 seconds of transfer with very few deviations from this value contrary to the baseline where the instantaneous throughput is varying a lot during the transfer.

The most noticeable result concerns the RTT and the jitter, with a reduced mean

RTT from 55 to 32 ms and a standard deviation of delay from 25 to 3 ms. The RTT is very stable after the BBR has converged at second 3 as plotted in Fig. 6.7d compared to Fig. 6.7b. The RTT reduction is for a part the result of a reduced uplink OWD (uplink ACK flow is similar to a CBR traffic) but for a bigger to the reduction the downlink queueing latency. Indeed, the average DL RLC buffer size was equal to 292.2 kB in the baseline but 186.6 kB (−36%) with eBSR. The uplink jitter reduction has a beneficial on the slow start phase of TCP that was identified as an issue in cellular networks (Chap. 5 with the network capacity hit in 198 ms instead of 259 ms, a difference of 2 RAN RTT). Also, the first megabyte is transmitted quickly in 282 ms instead of 340 ms, a reduction of 17%.

Results on TCP Cubic with hystart should be more important. Preliminary results give a median throughput on the first megabyte at 12 Mbps compared to the 7 Mbps found in Fig. 5.6. It is in accordance with observations concerning impact of slow-start early exit in [207].

## 6.4 Conclusion and Further studies

We have proposed a 3GPP-compliant modification in the base station's MAC scheduler to estimate UE's uplink transmission buffer status. The proposed algorithm takes into account the standardized SR and BSR signalling, but also the transport block utilization and grants, to perform a continuous estimation of UE's transmission needs.

In a lab testbed using an OpenAirInterface LTE BS we demonstrate the ability of the proposed solution to estimate and predict UE's transmission needs with as a result a reduction of uplink latency. The uplink traffic profile reveals to be less bursty, which benefits TCP downlink transmissions with better throughputs obtained and greater stability in the experienced RTT.

The main **limitations** reported by the experimental evaluation are:

— The difficulty to handle efficiently bursty non-deterministic traffics;
— The convergence time of estimation $\hat{D}$ towards $D$ impacting performance for short and sporadic traffic;
— The dependence on uplink transmissions occurrence to update $\rho$;
— The un-differentiation of dataflows with different packet generation characteristics in the uplink traffic mix.

We see 4 further studies for future validation of the proposition.

A more comprehensive evaluation with a **more realistic air interface** and user traffic should be conducted to fully validate the estimation algorithm, especially to study interactions with the endpoints and scheduling algorithm when users are in competition for radio resources with varying radio capacities.

A more comprehensive evaluation on a **wider range of traffics**, for instance with TCP Cubic (that continue to be widely common on the internet), video live-streaming (DASH protocol) and cloud gaming. TCP Cubic and cloud gaming especially should benefit from a more stable uplink latency.

The model when enabled could be the target of malicious users. It raises **security considerations** at this sensible point of the network. For instance, as the model continuously estimate the UE needs in transmission, a user could generate important traffic that overcome the cell's capacities. Such attack works for a scheduling algorithm that does not guarantee the fairness between users, with the consequence to disturb connections in cell area for the other users.

The TCP downlink transfer experiment revealed in paragraph 6.3.2.3 the **interaction**

**of the TCP feedback loop** to sense the network capacities with the enhanced-BSR loop to sense the uplink application transmission needs. The increase of the transmission datarate in the downlink should as a result increase the acknowledgement datarate in the uplink that is estimated by the enhanced-BSR estimation model. We think that this aspect should be investigated as well as the contribution to the reduction of the bufferbloat because we observed during experiments a reduction of DL buffering with both BBR and Cubic.

Chapter 7

# Towards an uplink scheduler combining latency, throughput and energy efficiency

**Abstract** *In this chapter, we envision a research perspective for Enhanced-BSR with the development of an uplink scheduling algorithm which takes advantage of Enhanced-BSR traffic prediction and Fast-UL to reduce latency, increase radio resource utilization in the cell while containing the energy consumption. We propose a Frequency-Domain (FD) scheduling algorithm which reveals good properties compared to the baseline Proportional-Fair (PF) scheduler in preliminary evaluations performed in a simulator.*

## Contents

**7.1 Motivation . . . . . 156**
**7.2 Radio resource allocation . . . . . 156**
7.2.1 Scheduling algorithm . . . . . 156
7.2.2 Time-domain scheduling . . . . . 157
7.2.3 Frequency-domain scheduling . . . . . 158
**7.3 Energy efficiency considerations . . . . . 159**
**7.4 Proposed uplink scheduling method using traffic prediction . 159**
7.4.1 Adaptive periodicity of transmission opportunities . . . . . 160
7.4.2 Distribution of terminals over frames . . . . . 162
7.4.3 Link adaptation and fairness with Proportional-Fair . . . . . 162
**7.5 Preliminary Evaluation . . . . . 163**
7.5.1 Simulation setup . . . . . 163
7.5.2 Performance analysis . . . . . 165
**7.6 Summary . . . . . 167**

## 7.1 Motivation

In the development of *enhanced-BSR* in Chapter 6, the uplink dynamic scheduler has not been modified. In this configuration, UG transmissions frequency is directly linked to SR transmission periodicity (see Fig. 6.5b). When we disable explicit SR, the continuous estimation of buffer status leads to a persistent scheduling of the UE every TTI resulting in a dreadful battery consumption.

We propose in this chapter to work on an uplink scheduling algorithm which tries to reconcile: 1) the latency jitter; 2) the radio resource utilization; 3) the energy consumption. Latency, achieved throughput and energy consumption constitute a fundamental trade-off in radio networks [13, 199, 248]. In this work, the usage of *enhanced-BSR* and *Fast-UL* makes the uplink scheduling more similar to the downlink scheduling. A particular attention is also put on achieved throughput, experienced latency and inter-UE fairness to comply with BE context.

## 7.2 Radio resource allocation

### 7.2.1 Scheduling algorithm

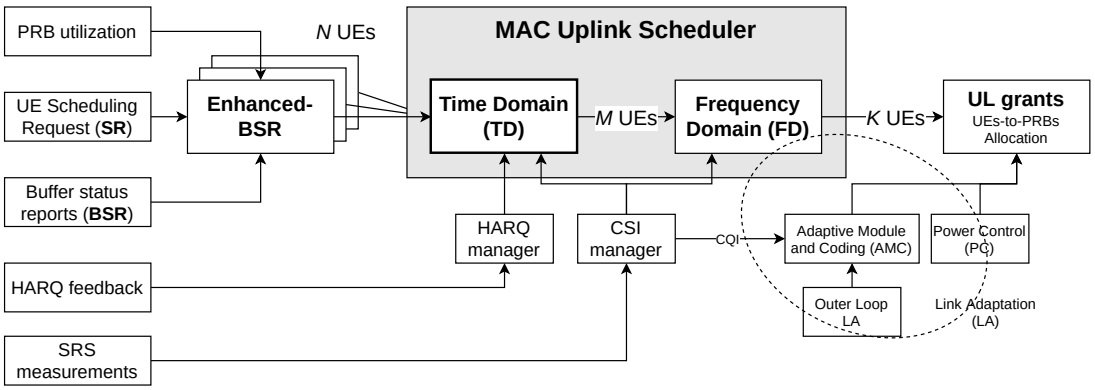



Figure 7.1 – Uplink dynamic scheduling model including enhanced-BSR prediction.

The aim of the BS MAC scheduler is to assign to each user PRBs while respecting a certain number of constraints due to the air interface with an objective of maximizing radio resource usage. Grant-based method is preferred for BE users (without QoS constraints) in place of grant-free access. Grant-based access is characterized by the usage of UG to resolve the contention problem of the shared medium between emitting UEs. UG is the

exclusive right given to a scheduled UE $i$ to transmit over a PRB $k$ on a TTI $t$ with a given Modulation and Coding Schemes. Grant-free accesses are not considered in this work since they are not designed for BE traffics. The UG is the outcome of the BS MAC scheduling process as illustrated in Fig. 7.1. The optimization challenge of mapping UEs to RBs comes with a prohibitive complexity with an exhaustive search.

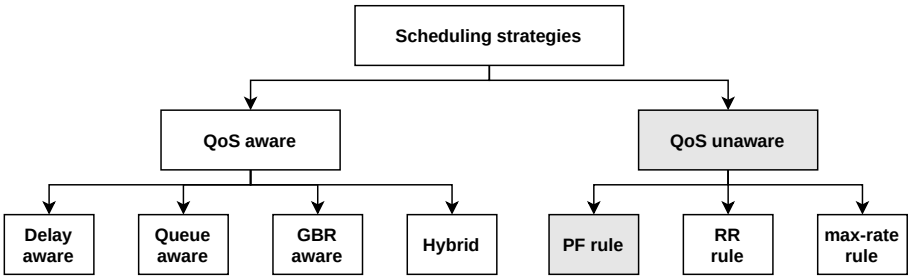



Figure 7.2 – Classification of scheduling strategies.

Multiple strategies have been proposed in the literature to solve scheduling problem, especially for the downlink side [57, 249] even if the uplink scheduling is more and more considered with the raise of uplink traffic and services [58, 59, 250]. We distinguish two families of scheduling algorithm [54] depicted in Fig. 7.2, those QoS-aware which take into account QCI for QoS requirements and those QoS-unaware mostly in use for BE. QoS-unaware scheduling strategies are of 3 families [52]. Round-Robin (RR) rules optimize fair sharing of radio resources between users. Best-CQI / Max-Rate (BCQI) or *max-rate* optimizes cell aggregated throughput even if it implies to starve some users to serve another. PF algorithm optimizes throughput while maintaining fair share of radio resources between users.

A practical breakdown of the radio resource allocation problem is to consider two sub-problems [251]. The first one selects a set of UEs $\mathcal{U}$ that could transmit data in the TTI $t$. The next one performs the mapping between $\mathcal{U}$ and the set of RB $\mathcal{K}$ of the given TTI $t$. We call them respectively, Time-Domain (TD) and FD scheduling since TD selects UEs scheduled for a given time slot and FD selects time-frequency resource blocks to UEs scheduled.

### 7.2.2 Time-domain scheduling

Time-Domain scheduling is usually linked to SR access method (called in the following "explicit-SR") in grant-based access (see Figure 1.4). TD distributes connected UEs

between TTIs. Reception of a SR usually causes the scheduling of the UE for the next TTI by the TD just like the reception of a HARQ NACK. The buffer-aware scheduler also takes into account the estimated buffer to select UEs. However, it has to be noted that TD has free rein to select users, even they do not have request access as soon as it is connected and active (not DRX inactive). Fast-UL are typically grants issued by the TD not in response to a SR (see Fig. 1.4).

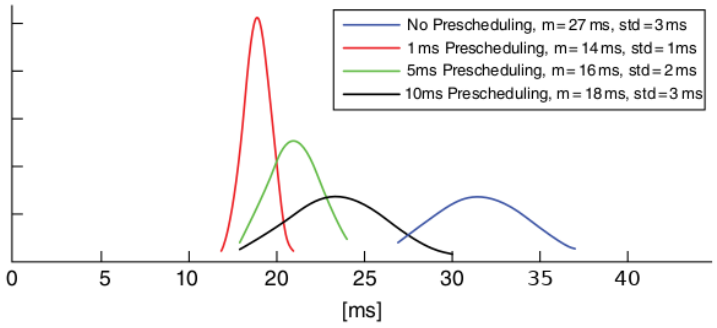



Figure 7.3 – Empirical distribution function of uplink latency for different values of Fast-UL periodicity. (Extracted from [26])

Fast-UL (or pre-scheduling) has been identified as an efficient solution to reduce uplink access latency [208]. Figure 7.3 extracted from [26] shows the beneficial impact on latency of regular Fast-UL. The smaller is the Fast-UL periodicity, the lower are the latency and the jitter. The results for the pre-scheduling of 1 ms is similar to what we got in short SR experiment in §4.2.4.1.

### 7.2.3 Frequency-domain scheduling

FD scheduler is always closely linked to Link Adaptation (LA) stage of MAC layer which attributes MCS to an UG as shown in Fig. 7.1. The frequency domain scheduler presented and proposed in the following is channel-aware, i.e. the scheduling decision is also based on capacities achievable by each UE. The literature generally considers the scheduling problem as a problem of maximization of a given metric $\lambda$ under a set of constraints. QoS-unaware schedulers usually choose a metric $\lambda$ related to datarate, to select the user $i$ that will get the PRB $k$ in maximizing $\lambda_{i,k}$. For instance, the fair-sharing of radio resources between terminals is a common constraint in BE context.

## 7.3 Energy efficiency considerations

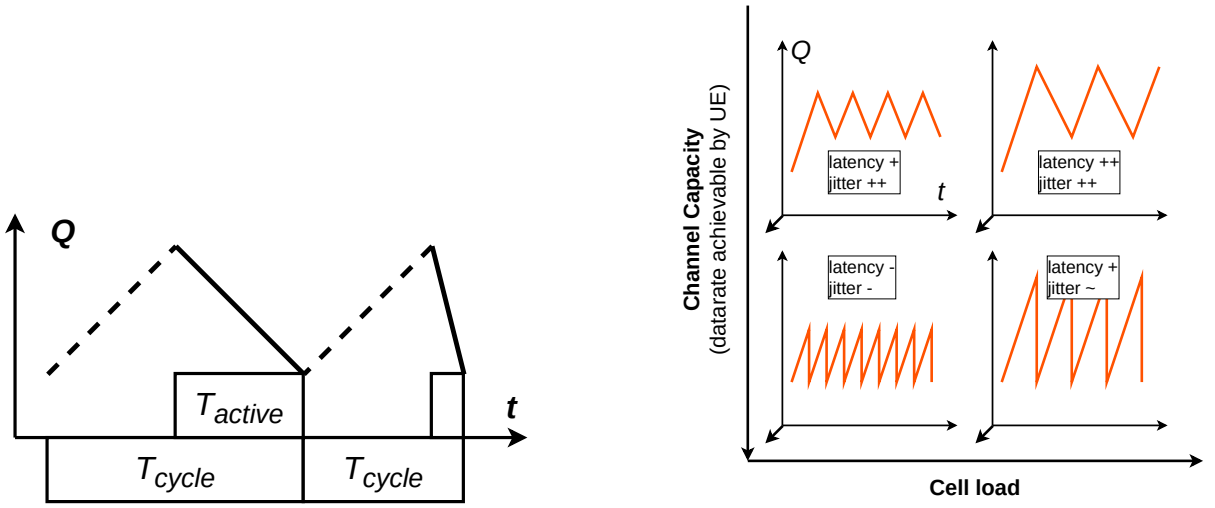
The transmission in the uplink is battery consuming for the UE [62]. More than in downlink, uplink transmissions cost in terms of energy for the terminal since it must transmit with a defined power to optimize channel SNR. We use the energy model proposed by Lauridsen et al. in [199]-2. This energy model describes the energy consumption of device's LTE modem. Because we consider only uplink, the power consumption $P$ ([Watt]) is defined by:

$$\begin{aligned} P = & m_{idle} \cdot P_{idle} + \\ & \bar{m_{idle}} \cdot \{P_{con} + m_{Tx} \cdot m_{Rx} \cdot P_{Rx+Tx} + \\ & m_{Tx} \cdot [P_{Tx} + P_{TxRF}(\mathcal{P}) + P_{TxBB}(d)]\} \end{aligned} \tag{7.1}$$

where $m_x$ corresponds to the share of time spent in the $x$ state; $P_{idle}$ the power associated to idle state when UE is connected but not active; $P_{con}$ the baseline energy consumes when modem is on; $P_{Rx+Tx}$ is the power to send or receive a signal; $P_{Tx}$ the baseline power for transmission module; $P_{TxRF}$ the energy consumed to transmit with a power $\mathcal{P}$ ([dBm]) and $P_{TxBB}$ the energy consumed to transmit at a datarate $d$ (with $d_t = g_t/T_{\text{tti}}$ [Mbps]). Parameters $P_x$ used for simulation are extracted from [252] and updated for 5G in [199]-6. Transmit power $\mathcal{P}$ depends on the channel conditions. $P_{TxRF}$ is constant for $\mathcal{P} \lesssim 0$ and increases exponentially for $\mathcal{P} > 0$ dBm. $P_{TxBB}$ is almost constant for lower and higher MCS, then datarate at which we transmit does not have a big impact on the power consumption. See in Appendix A.3 the typical energy consumption of a device over time from power up to data transmission. The major limitation of this energy model is that it does not integrate the RACH procedure and the DRX mode for a TTI-level computation of energy. However, the scheduler does not operate during RACH and DRX sleep. Thus, we are able to estimate energy consumption at each TTI using the Equation 7.1.

## 7.4 Proposed uplink scheduling method using traffic prediction

The proposed algorithm consists mainly to select which users has to be scheduled in a given TTI $t$. The important idea of this proposition is to use the principle of pre-




(a) Transmission pattern over time with cycle of $T_{cycle}$ length, with transmission ON for $T_{active}$ ms.

(b) Sawtooth transmission pattern according to the channel capacity (related to a UE) and cell load (related to the RAN).

Figure 7.4 – Adaptive periodicity of transmission opportunities pattern

scheduling and eBSR traffic prediction to reduce latency and jitter while preserving energy consumption and radio resource utilization. The key element is Fast-ULs with an adapted periodicity according to enhanced-BSR traffic prediction and transmission history.

The adaptation of Transmission Opportunity (TxOp) periodicity is presented in the next section §7.4.1. The radio resource utilization goes through a distribution of terminals over several time slots (§7.4.2). Inter-UE fairness is ensured by the usage of a PF FD which will be discussed in §7.4.3. The notations are summarized in the table of list of symbols in Appendix.

### 7.4.1 Adaptive periodicity of transmission opportunities

According to [199, 248], the uplink transmission energy consumption is reduced by: (L1) Minimizing the number of users transmitting in the same TTI; (L2) For a given mean throughput, it is preferred to transmit on a short period of time but with a high instantaneous throughput; (L3) The energy cost to turn on the modem is fixed. Thus, to minimize energy the number of "turn on" instructions should be reduced. It is preferred to transmit during 3 TTIs in a row instead of one out of two during 6 TTIs.

Starting from the above constraints, we propose to use a "cycle" pattern for transmission as shown in Figure 7.4a. During $T^{i,cycle} - T^{i,act}$ TTIs, no UGs are sent to UE

$i$: its transmission buffer $Q^i$ is filling. After that, during $T^{i,act}$ one or multiple UGs are sent in accordance with (L3) to empty the buffer at the highest rate to minimize duty cycle $\frac{T^{i,act}}{T^{i,cycle}}$ (L2). Contrary to pre-scheduling where usually $T^{act} = 1$ and $T^{cycle}$ fixed, we propose to update them at the beginning of every cycle as follows:

$$T_t^{i,act} = \lceil \hat{Q}^i_{t+T_{\text{sched}}+T_{\text{tx}}} / d_t^i \rceil \tag{7.2}$$

That corresponds to the number of TTIs necessary to empty the predicted buffer if all the bandwidth is given to the UE $i$. Thus, the period of time during which the UE is active depends on buffer length and channel capacity for UE $i$. Thereafter, the $T^{i,cycle}$ update is given by:

$$T_t^{i,cycle} = \frac{(T_t^{i,act} \cdot \bar{d}^i \cdot T_{\text{tti}}) + (\hat{D}^i \cdot T_{\text{tti}}) - \hat{Q}_t^i}{\hat{D}^i \cdot T_{\text{tti}}} \tag{7.3}$$

$$s.t.$$

$$T_{min} \leq \lceil T_t^{i,cycle} \rceil \leq T_{max} \tag{7.4}$$

$$T_{min} = T_{\text{harq}} \tag{7.5}$$

$$T_{max} = \lceil T_{\text{harq}} \cdot f(\bar{U}^{act}) \rceil \tag{7.6}$$

Equation 7.3 means the cycle length should be long enough to empty in $T^{act}$ TTIs the buffer generated during the cycle period $T^{cycle}$ and the remaining buffer from the previous cycle. It depends on estimated uplink datarate, channel capacity, and buffer size. $T^{cycle}$ is lower bounded by (7.5), the feedback loop delay for retransmissions and (7.6) to avoid too big transmission bursts. The choice to bound (7.5) to $T_{\text{harq}}$ is justified by the fact that if an HARQ retransmission, the new transmission will coincide with the retransmission. $\bar{U}^{act}$ in (7.6) corresponds to the average number of UEs served by the BS and is linked to the cell load. Cell load $\xi_{\text{cell}}$ is typically defined as percentage of used PRBs vs available PRBs. $f$ is a function to be determined, for instance a log-like function depending on the number of actively transmitting UEs in the cell.

A consequence of our proposal is the generation of a "sawtooth" pattern illustrated in Fig. 7.4b. According to cell load (if there is inter-UEs competition for access to radio resources) and channel capacity, the sawtooth pattern changes with the cycle length and the burst size. The best case occurs 1) when channel capacity permits the transmission of all the buffer in one TTI at a maximum datarate and 2) a low cell load with low

concurrency with other UEs for access to resources. Conversely, the worst case occurs when there are more UEs in the cell and when the channel capacity for the UE is low.

### 7.4.2 Distribution of terminals over frames

Following (L1) constraints the algorithm tries to minimize the number of UEs active on a TTI. At the update of the couple ($T^{i,act}$, $T^{i,cycle}$), we also update the beginning slot of the cycle with a shift of $\Delta t$ such as:

$$\arg\min_{\Delta t}\{\sum_{\tau=\Delta t+t+T^{i,cycle}-T^{i,act}}^{\Delta t+t+T^{i,cycle}} U_{\tau}^{act}\}, \quad \Delta t \in [t; t+T^{i,cycle}] \tag{7.7}$$

Where $U_{\tau}^{act}$ is the number of active users in the TTI $\tau$. In other words, the objective is to find where to shift the beginning of the transmission cycle such that the average number of active users in a TTI is minimized. In addition, distributing active users over radio frame is beneficial to frequency-domain scheduling to achieve fairness between them while decreasing complexity and increasing resource usage.

### 7.4.3 Link adaptation and fairness with Proportional-Fair

We use PF as frequency-domain scheduling strategy because it has good radio resource usage performances, and it is fair by-design [52]-Table 2. At each scheduling round $t$, the algorithm selects for each PRBs $k$ in $\mathcal{K}$ the UE $i$ which maximizes the metric $\lambda$:

$$\max_{i \in \mathcal{U}_{act}} \lambda_{i,k} \tag{7.8}$$

$$with \qquad \lambda_{i,k} = \frac{d_k^i(t)}{\bar{D}_t^i} \tag{7.9}$$

$$and \qquad \bar{D}_t^i = (1-\beta)\bar{R}^i(t-1) + \beta d_t^i, \quad \beta = 0.1 \tag{7.10}$$

Where $d_t^{i,k}$ is the achievable throughput for the UE $i$ over PRBs $k$ and $\bar{D}_t^i$ the mean achieved throughput including throughput already scheduled $d_t^i$ in the current TTIs $t$. PF is channel-aware at PRB scale but not buffer-aware. Link adaptation and physical constraints are not studied in this work.

A back-pressure is applied by PF on the TD scheduler through the $\hat{Q}^i$ value. In fact, for $\sum_{t=1}^{T^{i,act}} d_t^i < \hat{D}^i \cdot T^{i,cycle}$ (with $d^i(n)$ scheduled bytes in the TTIs $t$), it comes a rise of

$\hat{Q}^i$, that will be taken into account to update active and cycle duration. $d_t^i < d_t^{i,\mathcal{K}}$ when the bandwidth is shared between multiple TTIs, and it is the result of FD scheduling.

The complexity of proposed scheduling method is linear in the number of users and PRBs. It is similar to PF solutions with a prediction in a short future that comes with extra complexity to plan over multiple TTIs. The computation overhead of ($T^{act}$, $T^{cycle}$) is low and comes from overhead due to SRS-to-CQI and *enhanced-BSR* model processes.

## 7.5 Preliminary Evaluation

### 7.5.1 Simulation setup

A strong part of this work focus on the dynamic of radio resource allocation in uplink channel. The OAI-based lab experiment has a central role into understanding and assessment of proposed methods. However, we were confronted with the complexity of development process inside OAI code and instabilities related to it. For that reason, we set up a simulation platform to accelerate solution development on a simplified model of uplink radio allocation. The simulator, written in Python, is simplified to only simulated uplink radio resource allocation part with an emulated variable air interface. More details about the *alloc_sim* simulator are given in appendix F.7.6. Especially, the cellular network system model used is illustrated in appendix F.F.1 but it uses the design presented in Figure 1.5. The system model focuses on the uplink allocation with 4 elements:

1. The uplink traffic generator
2. The radio resource control plane
3. The uplink radio channel
4. The buffer-aware, channel-aware MAC scheduler

**Uplink traffic generator** The uplink traffic generation is the same as for OAI testbed presented in the research method section 2.3. The simulator manipulates real IP packets with the linux iptable *nfqueue*[1] (see details in Appendix F.7.6).

**Radio resource control plane** The control-plane mainly consists in signals used for grant-based radio resource allocation, SR, BSR and UG. The grant-based allocation scheme is described in §1.1.3.2. We use the same implementation as in OAI and srsLTE

1. `https://home.regit.org/netfilter-en/using-nfqueue-and-libnetfilter_queue/`

studied in §4.2.3. We also implement pre-scheduling with fixed periodicity and SPS for comparison purposes. Unless otherwise stated, the configurations used are those presented in Table 2.2.

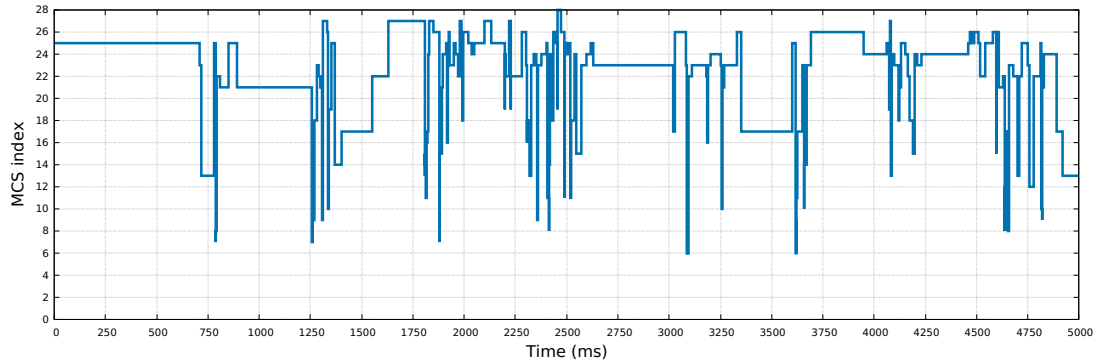



Figure 7.5 – Example of MCS traces from a commercial network used for the air interface simulation.

**Uplink radio channel** The uplink radio channel is updated every TTI. We use traces from the uplink extracted from an LTE commercial network to have realistic radio channel capacity variation. Traces indicate TBS and MCS extracted from Downlink Control Information (DCI) associated to a UG (which is easier to use than CQIs). The MCS and TBS indicates the instantaneous achieved throughput $d_t$. An example of trace is plotted in Fig 7.5 and exhibits an important variation of the MCS at small-time scale. The dataset is composed of 2060 traces sorted by RNTI value with in average 2000 UGs.

Radio interface loss is set randomly at the transmission block level. At any error, the block is considered as faulty and induces a retransmission 8 ms later. The generally targeted Block Error Rate (BLER) for the selection of the MCS and the transmission power is 10%, the value we use in the simulations.

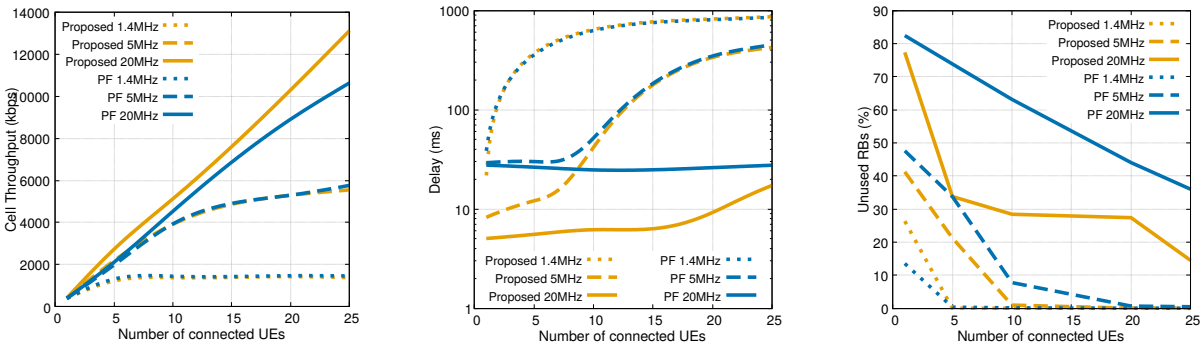
**MAC scheduler** MAC scheduler follows the model presented in §1.1.3.2 and in Fig.1.5. Each TTI the uplink scheduler calculates the radio resource grid for incoming time slots. It is buffer-aware as it uses the estimated buffer size to generate grants. We implemented the OAI model of estimation (§4.2.3) and enhanced-BSR. It is also channel-aware as it implements a link adaptation algorithm (linked to the channel capacity from commercial traces). We implemented 3 QoS-unaware algorithm (those described in §7.2.1), PF, RR and BCQI for comparison purposes. However in the next section, we will only consider

the PF algorithm because it is the most common strategy to ensure both fairness and throughput performances in the cell. Comparisons with others QoS-unaware scheduling algorithms are let for a further work.

### 7.5.2 Performance analysis

| **Configuration** | **Cell throughput** (kbps) | **Unused RBs** (%) | **Mean #user** per TTI | **Mean #TxOps** per UE | **Electric** Charge (mAh) | **Mean delay** (ms) | **Mean jitter** (ms) |
|---|---|---|---|---|---|---|---|
| 1.4 MHz / 1 UE | 393 / 397 | 21.1 / 44.0 | 0.27 / 0.25 | 545 / 494 | 1.37 / 1.29 | 32.0 / 16.1 | 8.2 / 3.1 |
| 1.4 MHz / 5 UEs | 1131 / 1149 | 0.3 / 0.0 | 4.22 / 3.69 | 1691 / 1477 | 1.13 / 1.14 | 444 / 424 | 247 / 240 |
| 5 MHz / 5 UEs | 1984 / 2065 | 32.0 / 25.5 | 1.0 / 0.7 | 403 / 296 | 2.28 / 1.92 | 29.16 / 10.82 | 9.03 / 6.77 |
| 5 MHz / 25 UEs | 5032 / 5073 | 0.39 / 0.0 | 18.87 / 15.86 | 1509 / 1258 | 0.96 / 0.96 | 422 / 492 | 238 / 275 |
| 20 MHz / 25 UEs | 10702 / 13111 | 33.0 / 14.8 | 4.58 / 2.88 | 362 / 232 | 2.54 / 2.89 | 27.0 / 16.38 | 9.30 / 13.17 |

Table 7.1 – Summarized uplink scheduling results. *baseline PF / proposed* for a simulation of 5 seconds.

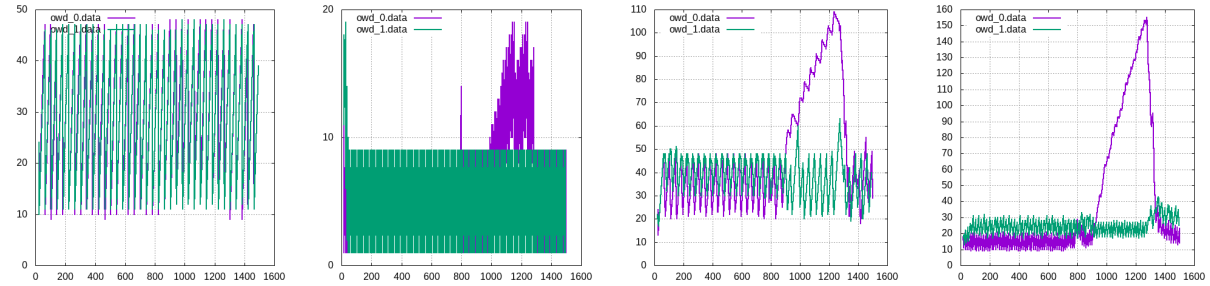



(a) Cell aggregated (b) Uplink delay (c) Unused RBs

Figure 7.6 – Uplink scheduling results for variable number of users and for different bandwidths.

In this section we provide a preliminary performance evaluation of the proposed uplink scheduling algorithm. We limit simulation to one cell with users randomly distributed in the cell. The packet arrival process for each user is 50 bytes/ms, corresponding to a deterministic CBR traffic at 400 kbps. BSR periodicity is set to 32 ms, SR periodicity to 10 ms and HARQ feedback loop to 8 ms. We compared our proposed TD scheduling algorithm to current explicit SR-based with the same PF FD scheduling. We vary the number of users from 1 to 25 users and the bandwidth from 1.4 to 20 MHz (6 RBs to 110). Simulation is performed for 5000 TTIs (=5 sec) and repeated 20 times. Figures 7.6 compare cell aggregated throughput $\bar{D}_{\text{cell}}$, mean uplink delay $\bar{T}$, and radio resource usage $\bar{\xi}_{\text{cell}}$ for different simulation parameters. Other overall results are summarized in Table 7.1.

It is clear that the proposed FD algorithm outperforms the baseline in terms of latency and jitter in all experiments. It also increases aggregated cell throughput for wide bandwidth congested cases thanks to a better radio resource usages. Wide bandwidths are becoming very common with 5G RAN when uplink channel will see its datarate increase in the incoming years. In the same time, energy consumption remains equivalent between the 2 methods by reducing number of Transmission Opportunity (TxOp)s (L2), (L3) and the number of average active users in a TTIs (L1). Our proposition optimizes the trade-off between throughput, latency and energy for PF FD scheduling without modifications of the existing signalling or QoS management.

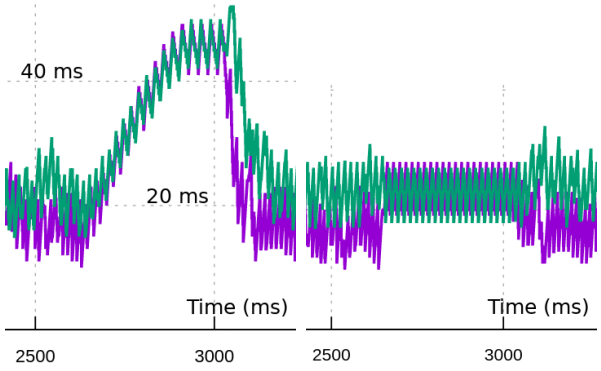



(a) Baseline PF / 10MHz (b) Proposed / 10MHz (c) Baseline PF / 1.4MHz (d) Proposed / 1.4MHz

Figure 7.7 – Uplink delay (ms) evolution over time according to scheduling algorithm (Proposition vs PF) and bandwidth.

In Figures 7.7 are depicted the uplink OWD evolution over 1.5 sec for 2 UEs for 10 MHz and 1.4 MHz. The evolution of CQI over time, retransmission events and uplink datarate are the same in all the cases. The first thing to observe is the reduction of latency thanks to *enhanced-BSR* prediction instead of BSR-32 ms estimation. For a wider bandwidth of 10 MHz (Fig. 7.7a and Fig. 7.7b), our algorithm achieves also a lower jitter while minimizing the number transmitting TTIs. Another good characteristic is the increase of experienced latency by $UE_0$ even if the bandwidth is enough wide to maintain a low latency when the capacity decreases. Such implicit signal of change in the radio environment is useful for applications to adapt datarate over time while with PF it was almost completely hidden. For Fig. 7.7c and Fig. 7.7d the bandwidth is only of 1.4 MHz. When the channel capacity decreases for $UE_0$ the latency increases quickly to a hundred milliseconds with PF and with our proposition, but the jitter remains contained. After TTI 1300, the $UE_0$'s channel quality recovers and the latency quickly decrease because the PF algorithm ensures a fair share of the throughput. During the recovery period,

PF prioritizes $UE_0$ over $UE_1$, implying a temporary decrease of achieved datarate and increase of latency and jitter for the $UE_1$ that could be the hint of inter-UE competition at the bottleneck for the endpoint.




(a) without distribution (b) with distribution of UEs over frames

Figure 7.8 – Uplink delay according to UEs distribution strategy.

We also evaluated the impact of terminals distribution over frames (Sec. 7.4.2, Figures 7.8) and found a clear local impact on spectral efficiency and latency when 2 UEs are in competitions during their active phase. Delayed from few milliseconds one UE to give all the bandwidth for 2 UEs in 2 different frames improves spectral efficiency and resulting in less latency and jitter in average. It also acts to reduce number of UEs active in a given TTIs.

## 7.6 Summary

This section presents some preliminary results on a research direction following Enhanced-BSR which is the uplink scheduler. We have proposed a QoS-unaware scheduling method which uses Enhanced-BSR traffic prediction and Fast-UL mechanism. The main idea is to issue UGs with a variable periodicity to minimize latency and jitter while maintaining radio resource usage, fairness and energy consumption. Simulation results show a clear improvement in terms of latency, radio resource usage and throughput compared to PF scheduling without any significant increase of energy consumption. There is still a lot of work to do, but preliminary results are promising for future developments. Further, it is important for a future publication to evaluate the proposed solution with state-of-the-art scheduling algorithms in a NS-3 simulation environment.

# Conclusion

## General conclusion

5G changes the current LTE paradigm to go towards low latency cellular networks. The low latency in cellular networks is often presented as a key enabler for emergent new internet services sensitive to latency such as cloud gaming, augmented reality, tactile internet. 5G-NR standardization integrates a number of radio transmission enhancements in order to reduce latency compared to the last generation of cellular network i.e. LTE. The RAN and end-to-end 5G toolbox, including a new flow management and QoS framework, is utilized and completed by an important research work to achieve the hard challenge of 1 ms latency in cellular networks. The main objective of this thesis was to reduce latency and jitter in the RAN.

In order to achieve this goal, we set up an experimental testbed using an OpenAirInterface base station and commercial devices with the objective to analyze RAN-related latencies. The radio channel conditions have been made perfect and stable in time to limit spurious HARQ retransmissions. The RAN RTT was measured with 2 passive measurement points and 1 active ICMP ping measurement. We have found limitations in measuring latency inside the BS where it is difficult to monitor latency of an individual packet inside the base station.

We then proposed a tool for fine-grain internal latency analysis inside the BS. This tool called *LatSeq* was designed to have a low impact on performance of the observed system. For that, we adopted a software architecture which ensures that latency-critical measurement part is made as fast as possible while the heavy analysis part is made offline. Data packets are observed during their journey inside the BS with measurement points that generate packet fingerprints. The packet fingerprint includes a microsecond-accurate timestamp, a position of the packet inside the base station and identifiers related to this packet. Position and identifiers are used to rebuild in offline the packet journey in the RAN. Packet journey is plotted in a waterfall format which is an easy and comprehensive tool for latency analysis.

After that, we performed a set of experiments to characterize latency inside the RAN

for a non-QoS internet traffic (i.e. Best-Effort). Especially, we illustrated the baseline RAN RTT under different configurations, the impact of RTT on HTTP transfer, and we exhibited the well-known bufferbloat phenomena which is the existence of a persistent queue inside the network bottleneck. The latency due to the bufferbloat represents in general 2 or 3 orders of magnitude that of the RAN. From these first experiments, we found an important latency and jitter caused by the uplink-channel. We have completed these observations with trials in a commercial mobile network and related observations in the literature. We have explained this latency and important jitter under ideal transmission conditions by the grant-based radio resource allocation scheme and the RAN configurations. According to the RAN configuration concerning the scheduling strategy, the SR periodicity and the BSR periodicity, the median RAN RTT obtained goes from 19 ms to 31 ms and the jitter from 3 ms to 13 ms. We have pointed out the importance of the BS knowledge of the UE buffer size for the scheduling performance, latency and jitter.

We motivated the need to tackle uplink latency jitter due to the bad estimation of the transmission buffer length with the example of the TCP performance in mobile networks. Indeed, we showed that TCP throughput suffers from rapid variation of the RTT measured with acknowledgment feedback loop. Especially, it leads to spurious RTO triggering and misestimate of the network congestion status, resulting in a datarate lowered than the available radio channel capacity. The problem still exists for the state-of-the-art of congestion control (i.e. BBR) and for Quic, the next generation of transport protocol.

We have proposed *Enhanced-BSR*, a new MAC layer module, which redefines the manner the UE buffer size estimation is made by the BS. For that, we have used a model that continuously estimates the buffer size that is then used by the BS uplink dynamic scheduler to allocate radio resources according to a scheduling strategy. The algorithm proposed in the first proposed version models the UE as a queue filled by an estimated CBR data source and emptied with UG. The estimation is corrected at the reception of a BSR and the transport block utilization following an UG. We have demonstrated the ability of the model to efficiently reduces latency and jitter in 3GPP LTE and 5G RAN without having resort to QoS bearer or further RAN configurations. We have shown the benefit of our solution on TCP transmissions with an increased throughput and a controlled end-to-end RTT.

Finally, this thesis includes a number of technical contributions. We have developed and implemented a new measurement tool inside OAI base station which requires an effort of software design to achieve stringent time requirements and a good knowledge of the

RAN protocol stack. LatSeq is intended to be integrated into the OAI open-source project. Also, LatSeq has been designed to be implemented in a wider range of project and thus, it has been the subject of a publication of the code in open-source. Enhanced-BSR was also developed and integrated in the OAI eNB. The code composed of 1750 source lines of code implement the linear model proposed. A patch of this code is available on request. Finally, we have developed a toy uplink radio channel to accelerate experimental development of radio resource allocation scheme and uplink dynamic scheduling algorithm. It is less representative than a NS-3 simulation, but the computational cost and the development time are clearly reduced for quick experiments.

# Perspectives

**Further latency studies** We have explained in the Sec. 4.2 a root cause of the uplink channel jitter comes from the partial knowledge of the UE's transmission buffer status that has the base station's scheduler. We began to study the different aspects and consequences of the partial buffer knowledge on the scheduling efficiency §4.2.5 and the transmission pattern §4.2.4. However, this work is still limited and should be completed by a comprehensive mathematical, analytical study, and simulations. Indeed, the majority of work concerning the uplink scheduling utilize as assumption a perfect knowledge of the buffer at a given TTI, like in the downlink. This assumption is false for a long BSR timer for the default BE bearer, thus relation between the scheduling algorithm and the partial buffer knowledge should be investigated.

The chapter 5 motivated the need to tackle uplink-channel jitter with the example of the TCP acknowledgments flow for downlink transmissions. In the same time, the Quic protocol is replacing TCP in number of mobile applications. The question of Quic parameters for mobile networks remains open in the IETF working group discussions [2] [3]. Especially, there are the open questions of ACK frequency [253] and return link for Quic in the mobile context. For instance, the study of latency jitter could bring arguments to the ACK frequency determination, with the trade-off between network capacity sensing and burst of transmission generation.

2. IETF Quic working group on Github : `https://github.com/quicwg`
3. IETF Quic working group mailing list archive : `https://mailarchive.ietf.org/arch/browse/quic/`

**LatSeq** We presented *LatSeq* in Chapter 3, a novel solution for internal latency measurement. LatSeq was implemented in the OAI eNB LTE stack for analyzing internal latency in the RAN (§3.3.2). We chose the LTE stack because the 5G-NR was not stable at this moment in 2020 and 2021. OAI 5G-NR stack is gaining stability in 2022 and LatSeq should get interest to be implemented for the gNB [4] Especially if it could be done in collaboration with the OAI community. Challenges around this implementation into the 5G stack are the same as presented in Sec. 3.3.2, i.e. relevant points for packet journey rebuilding and latency analysis. It is an ongoing work [5].

Research objectives behind LatSeq could be extended beyond the sole latency analysis on a testbed.

We see a first application to the high tail latency troubleshooting. Fig. 4.1 shows that for an extreme minority of packets, the RAN latency could be very important somewhere one magnitude above the median latency. The knowledge on the individual packet path inside the BS is a helpful tool for RAN troubleshooting. Indeed, it discriminates the source of the latency, BS or UE or backhaul, the protocol, etc. With the development of private 5G networks for enterprise [6] which need robustness and reliability, we hope to demonstrate the utility of LatSeq for network debuggability.

We envision a second application of LatSeq for latency monitoring of end-to-end Slicing with Service-level agreement (SLA). As said Roland Mestric, Nokia's head of product marketing [7], "*The SLAs really matter. These industries [...] run some critical business on [Slicing] services. That's where the slicing comes into play as a way to deliver this quality.*", the end-to-end slicing is associated to a QoS contractualized with SLAs. SLAs are based on numerical KPIs for instance maximum latency or limited burst of transmission. In this task, LatSeq with an adapted sampling rate (to limit the number of log data generated) can report latency KPI for packet journeys for a given slice. We have to demonstrate the advantage of LatSeq over BS reported latency values in particular, compared to ground truth and verifiability.

Furthermore, old architecture of BS used in 3G and LTE employs a monolithic system where all network functions of the BS are physically located at the bottom of a tower

4. 5G-NR BS L2 stack : `https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/develop/openair2/LAYER2`

5. `https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/latseq_5GSA`

6. `https://firecell.io/oai-for-private-5g/`

7. "Nokia announces cross-domain–core, transport and RAN–automated network slicing solutions", *RCRWirelessnews.com* (2020) : `https://www.rcrwireless.com/20201001/5g/for-4g-5g-network-slicing-slas-really-matter-nokia-says`

at the center of the cell, tower at the top of which the antennas are installed. A novel architecture initialized with LTE advanced and developed in 5G employs a decentralized, disaggregated architecture. The new gNB is composed of a Radio Unit (RU), a Distributed Unit (DU) and a Centralized Unit (CU). According to the split option in use, several network functions of CU are softwarized and centralized in a datacenter. C-RAN architecture [110, 112] is very flexible to adapt radio access architecture to the characteristics of the cell (e.g. rural or dense area) and operator's network infrastructure. LatSeq tool should take into account this architecture evolution and break up with its current monolithic architecture. This decentralized architecture raises issues of synchronization and data transfer. The interest of LatSeq for latency monitoring in these complex, decentralized and software-based infrastructures is certain.

In summary, the future of LatSeq has its place within 5G, C-RAN and slicing.

**Enhanced-BSR** The initial proposition of *Enhanced-BSR* in Chapter 6 uses a linear model of estimation for the traffic resource, which means the source is considered as CBR traffic with a uniform distribution of packet arrival. The model proves its efficiency for uplink CBR transfer, paced uplink TCP flow and TCP uplink ACK flows but, it is not well suited for short transmissions and bursty traffic patterns. In addition to that, the energy cost for the UE to be scheduled more regularly should be evaluated. However, according to the model of energy presented in §7.3 [199], the energy impact for the UE of a more frequent scheduling is compensated by a transmission at a higher datarate when the UE is transmitting (i.e. the UE modem is turn on to transmit over a larger set of PRBs).

The next step should explore the other estimation method to be adapted to a wider range of traffic patterns. This model of estimation has to use at least BSR sampling and TB utilization to provide BS MAC scheduler a continuous estimation of the UE transmission buffer length. There are already traffic pattern prediction algorithms proposed in the literature [236, 239, 254–257],[125]-4,[129]-7.2.1. for the context of radio resource allocation in the IoT and machine-to-machine communication. Online learning of the traffic pattern in a given cell with a Machine Learning techniques (e.g. [258] with Long Short-Term Memory (LSTM)) is an interesting approach for a more general context without knowing *a priori* the type of traffic to estimate in the cell.

Also, in the experiment, the linear model of eBSR was unique for all users in the cell, with the same configurations and for all cells using the OAI eNB program without taking

into account cell particularities (e.g. rural or urban context). It is clear that different models should be used in the same BS according to the expected traffic types associated to a bearer (it is hard to set up a specific eBSR model to a unique UE since the RNTI is likely to change in the time). Slicing and QoS identifier associated to it is a good indicator of the expected traffic pattern to adapt eBSR model and configurations in consequence.

So far, we have only talked about buffer size prediction with eBSR. The linear model we proposed also gives the estimated waiting time of the queue such as $\hat{W}_q = \hat{D} \times \hat{Q}$. This value could be used by a delay-based scheduler to minimize the latency of the last packet arrived into the buffer that is an efficient scheduling strategy to limit queueing delay (at the expense of aggregated cell throughput and inter-UE throughput fairness).

An important perspective of this research work was developed in the latter chapter, which is the development of a dynamic scheduling algorithm for the uplink channel. The objective of this thesis is to reduce latency in the RAN for BE traffic. The radio resource allocation scheme and strategy have a significant role on the RAN latency. The allocation strategy controlled by the scheduler have to take multiple trade-offs into consideration related to the radio transmissions. One of them is the trade-off between latency, throughput and energy efficiency. The scheduling algorithm we propose to develop preserves throughput and energy efficiency but have to decrease latency and jitter in the uplink. For that, it should benefit from estimations provided by the enhanced-BSR model, that is the key difference with the related work in the literature. Finally, the scheduler will have to ensure a certain throughput fairness between users in the context of BE enhanced Mobile BroadBand (eMBB). Milestones have been laid in the last chapter with encouraging preliminary solutions and results as a research perspective.

# APPENDICES

# A Background

## Initial Access

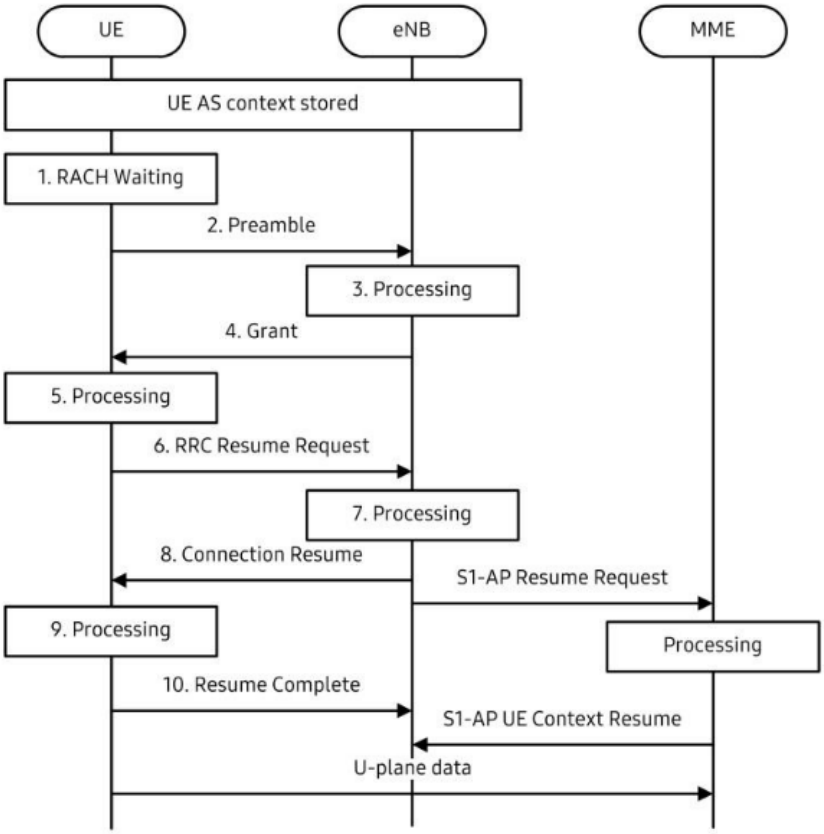



Figure A.1 – Random Access procedure.

5G implements new initial access procedure and RRC mechanisms with the aim to thin Control-Plane (CP) down and reduce CP latency.

Firstly, the initial access procedure by which a device finds a cell to connect with, receive the necessary system information and request connection through random access is made ultra-lean.

Secondly, RRC procedures have been shorten by reducing the number of steps [25]. The most notable evolution of RRC process in 5G is the introduction of a RRC "inactive" state between the Long Term Evolution (LTE)'s RRC "idle" and RRC "connected" states. In this state, no data transfer are possible but the RRC context and the connection to core network are kept established. In consequence of that, when a new data transfer

arrives, the RRC procedure is simplified to a "wake up" RRC message directly to the known Access Management Function (AMF) machine of the network core. The initial access latency is then clearly reduced in 5G, that should benefit for short flow duration and better energy efficiency.

## RRC and Control plane

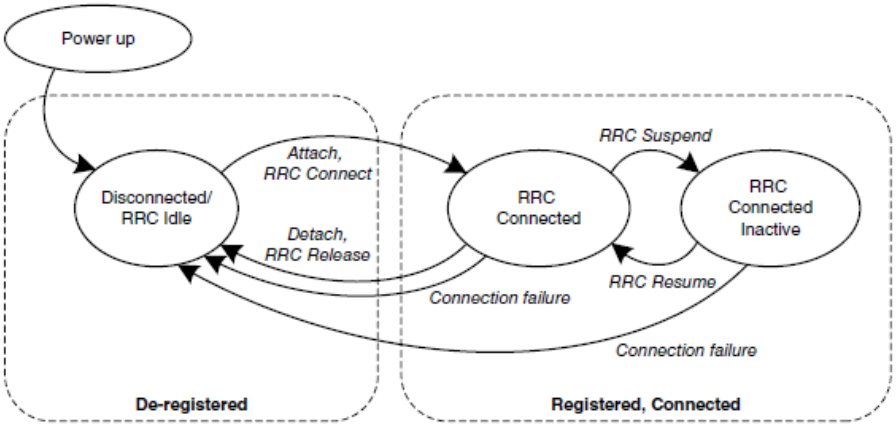



Figure A.2 – RRC state machine.

The RRC control-plane functionality operates between the RRC located in the BS. RRC is responsible for handling the RAN-related control-plane procedures. The device can be in different states depending on the traffic activity. In LTE, the disconnected and connected states are defined, 5G defines a new state for connected but inactive device.

## Energy consumption

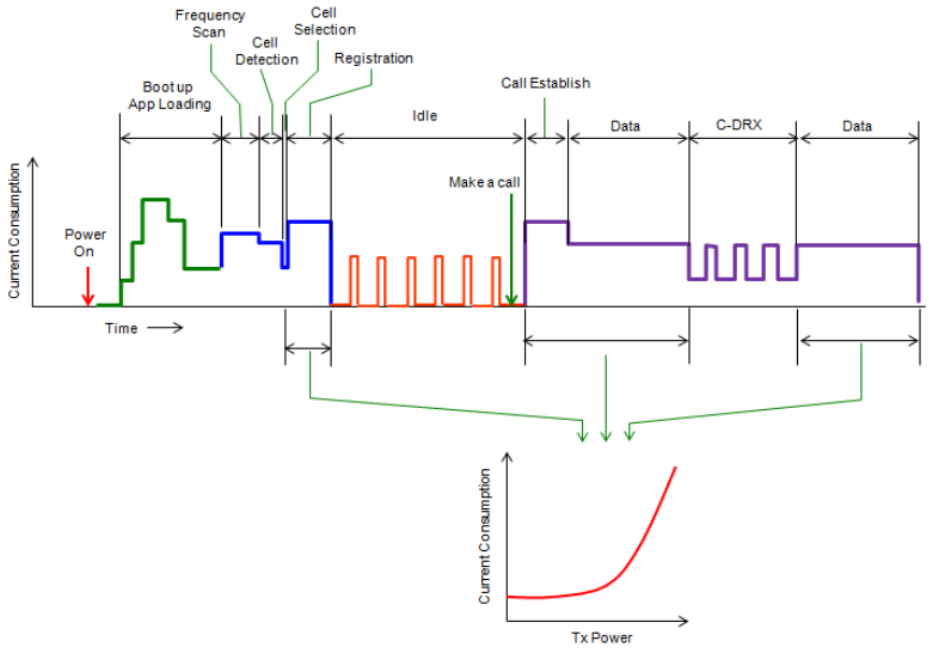



Figure A.3 – UE energy consumption over time *(Source: ShareTechnote.com)*

The current consumption of the device is really depends on the connection state and the transmission power.

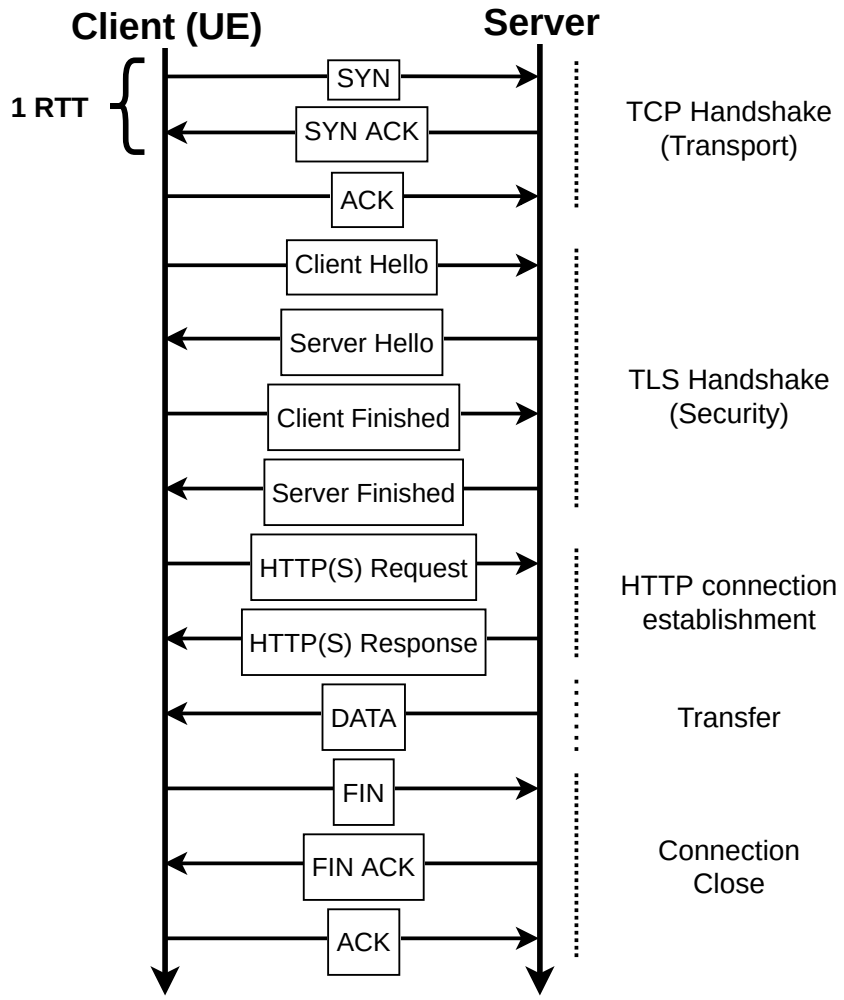



Figure A.4 – HTTP(S) request time-sequence.

# B Research Methods

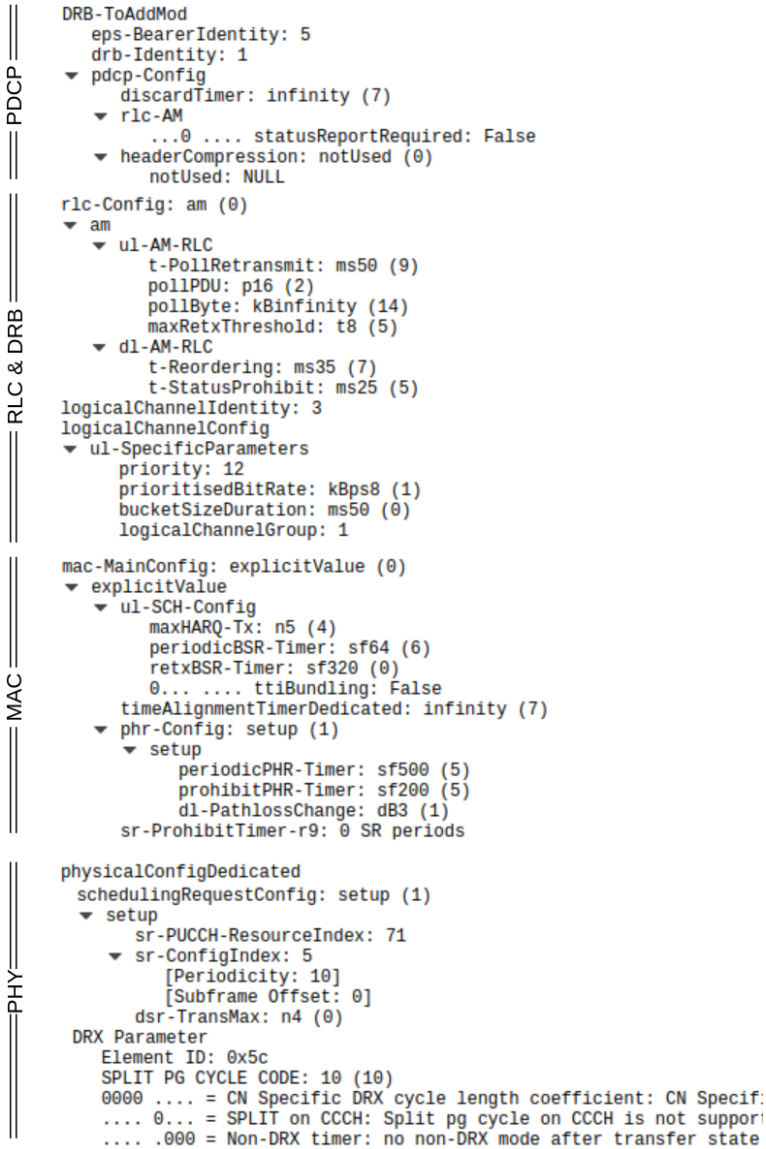



Figure B.1 – Data Radio Bearer configurations.

At the bearer establishment, a number of RAN-related parameters are initialized for PDCP, RLC, MAC and PHY layers. These parameters are shared between BS and UE to ensure the inter-operability.

# C LatSeq

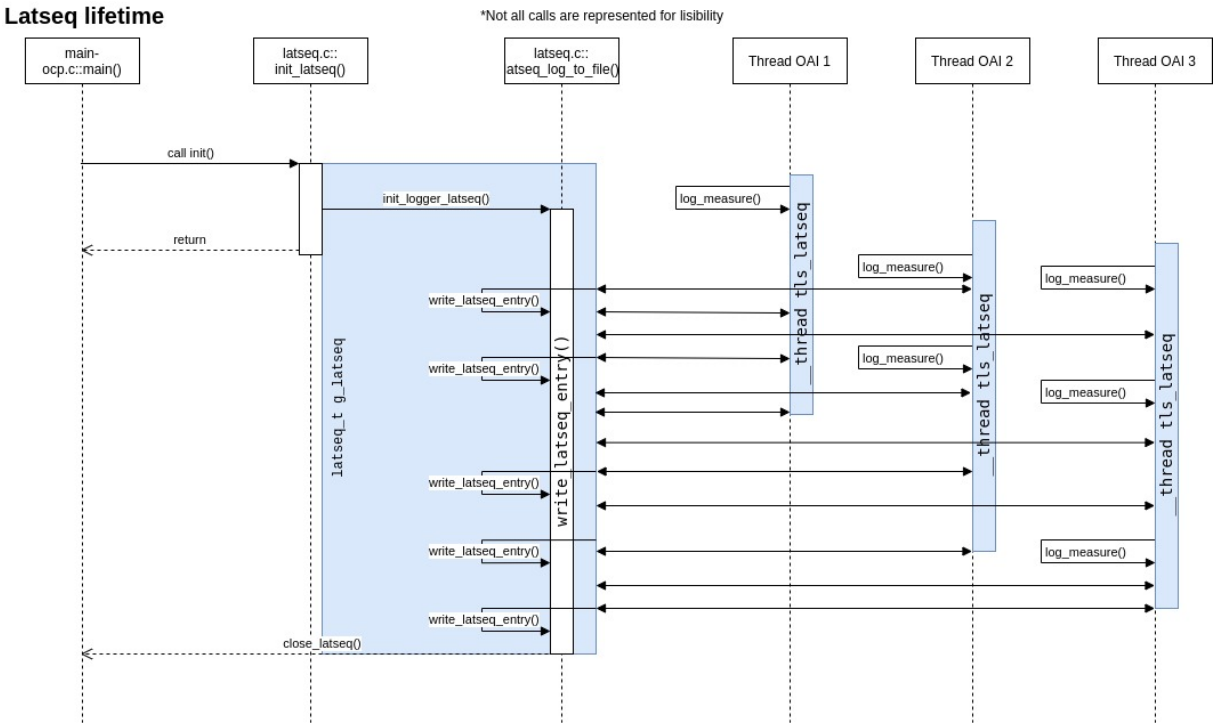



Figure C.1 – LatSeq measurement part lifetime.

```
Percent |    0000000000000f04 <log_measure2>:
        |    log_measure2():
        |    #endif
        |    */
        |    }
        |
        |    static __inline__ void log_measure2(const char * point, const char *fmt, uint32_t i1, uint32_t i2)
        |    {
  0,84  |      push   %rbp
  0,28  |      mov    %rsp,%rbp
  5,57  |      sub    $0x30,%rsp
  1,39  |      mov    %rdi,-0x18(%rbp)
        |      mov    %rsi,-0x20(%rbp)
  0,28  |      mov    %edx,-0x24(%rbp)
  3,90  |      mov    %ecx,-0x28(%rbp)
        |      if (tls_latseq.th_latseq_id == 0) {
  1,11  |      mov    $0xffffffffffffcff0,%rax
        |      movzbl %fs:(%rax),%eax
        |      test   %al,%al
  0,56  |    ↓ jne    32
        |        //is not initialized yet
        |        if (init_thread_for_latseq()) {
        |    → callq  init_thread_for_latseq
        |      test   %eax,%eax
        |    ↓ jne    d6
        |          return;
        |        }
        |      }
        |      latseq_element_t * e = &tls_latseq.log_buffer[tls_latseq.i_write_head%MAX_LOG_SIZE];
  4,18  | 32:  mov    $0xffffffffffffcff0,%rax
  1,39  |      mov    %fs:0x3008(%rax),%eax
        |      and    $0x7f,%eax
        |      mov    %eax,%edx
  4,18  |      mov    %rdx,%rax
  2,23  |      add    %rax,%rax
  0,84  |      add    %rdx,%rax
  4,18  |      shl    $0x5,%rax
  3,06  |      mov    %fs:0x0,%rcx
  1,39  |      mov    $0xffffffffffffcff0,%rdx
  0,56  |      add    %rcx,%rdx
  3,62  |      add    %rdx,%rax
  4,46  |      add    $0x8,%rax
  2,23  |      mov    %rax,-0x8(%rbp)
        |      e->ts = rdtsc();
  0,56  |    → callq  rdtsc
  2,51  |      mov    %rax,%rdx
  3,06  |      mov    -0x8(%rbp),%rax
  3,90  |      mov    %rdx,(%rax)
        |      e->point = point;
        |      mov    -0x8(%rbp),%rax
  0,56  |      mov    -0x18(%rbp),%rdx
  1,67  |      mov    %rdx,0x8(%rax)
        |      e->format = fmt;
  0,56  |      mov    -0x8(%rbp),%rax
        |      mov    -0x20(%rbp),%rdx
  5,01  |      mov    %rdx,0x10(%rax)
        |      e->len_id = 2;
  1,39  |      mov    -0x8(%rbp),%rax
  3,90  |      movw   $0x2,0x18(%rax)
        |      e->data_id[0] = i1;
        |      mov    -0x8(%rbp),%rax
  3,90  |      mov    -0x24(%rbp),%edx
  5,01  |      mov    %edx,0x1c(%rax)
        |      e->data_id[1] = i2;
  2,79  |      mov    -0x8(%rbp),%rax
        |      mov    -0x28(%rbp),%edx
  3,62  |      mov    %edx,0x20(%rax)
        |      tls_latseq.i_write_head++;
  1,67  |      mov    $0xffffffffffffcff0,%rax
  1,11  |      mov    %fs:0x3008(%rax),%eax
        |      lea    0x1(%rax),%edx
  2,51  |      mov    $0xffffffffffffcff0,%rax
  3,06  |      mov    %edx,%fs:0x3008(%rax)
  0,28  |    ↓ jmp    d7
        |          return;
        | d6:  nop
        |    }
  6,13  | d7:  leaveq
  0,56  |    ← retq
```

Figure C.2 – Assembly code of a LatSeq measurement point.

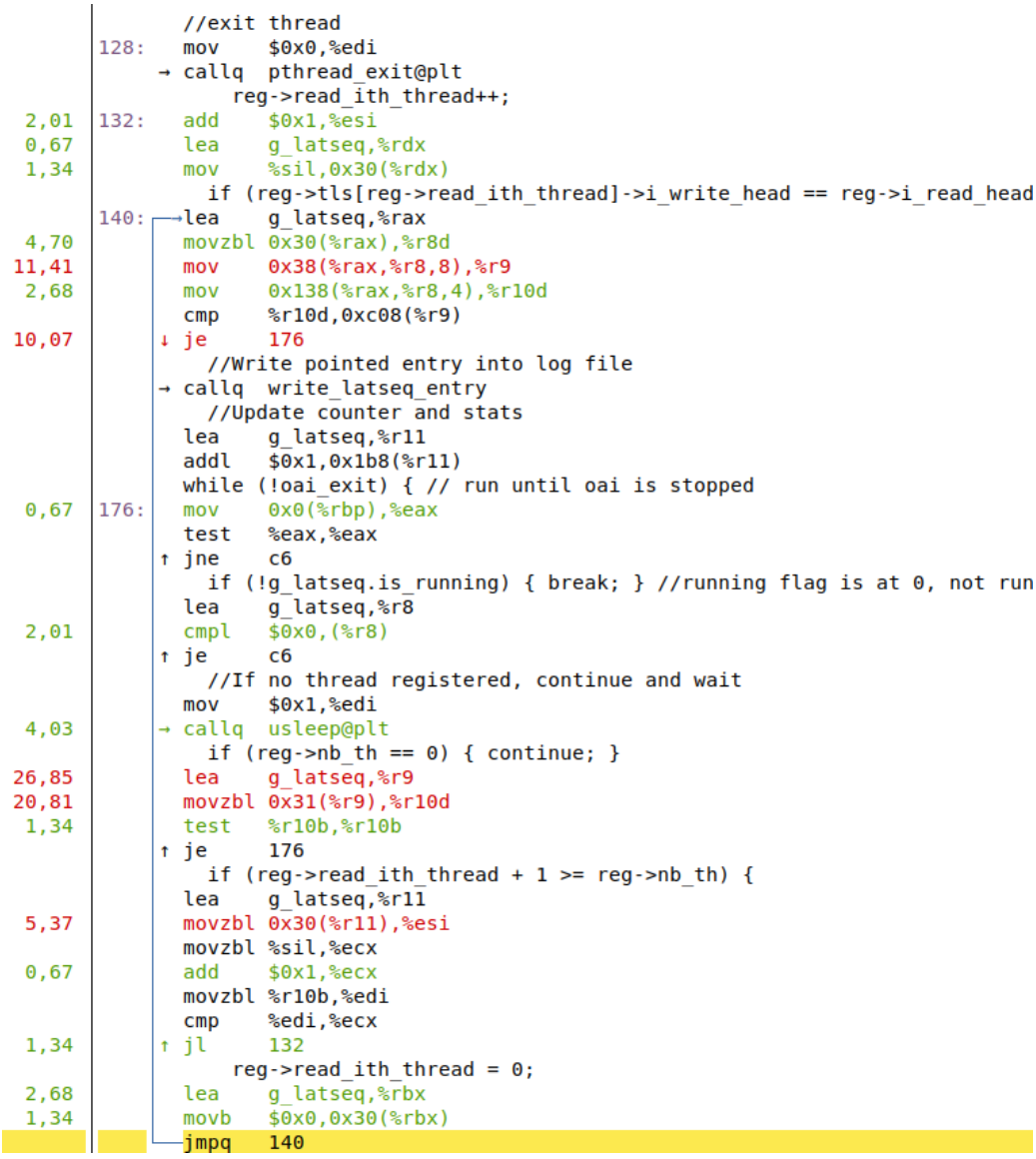



Figure C.3 – Assembly code of a LatSeq data collector.

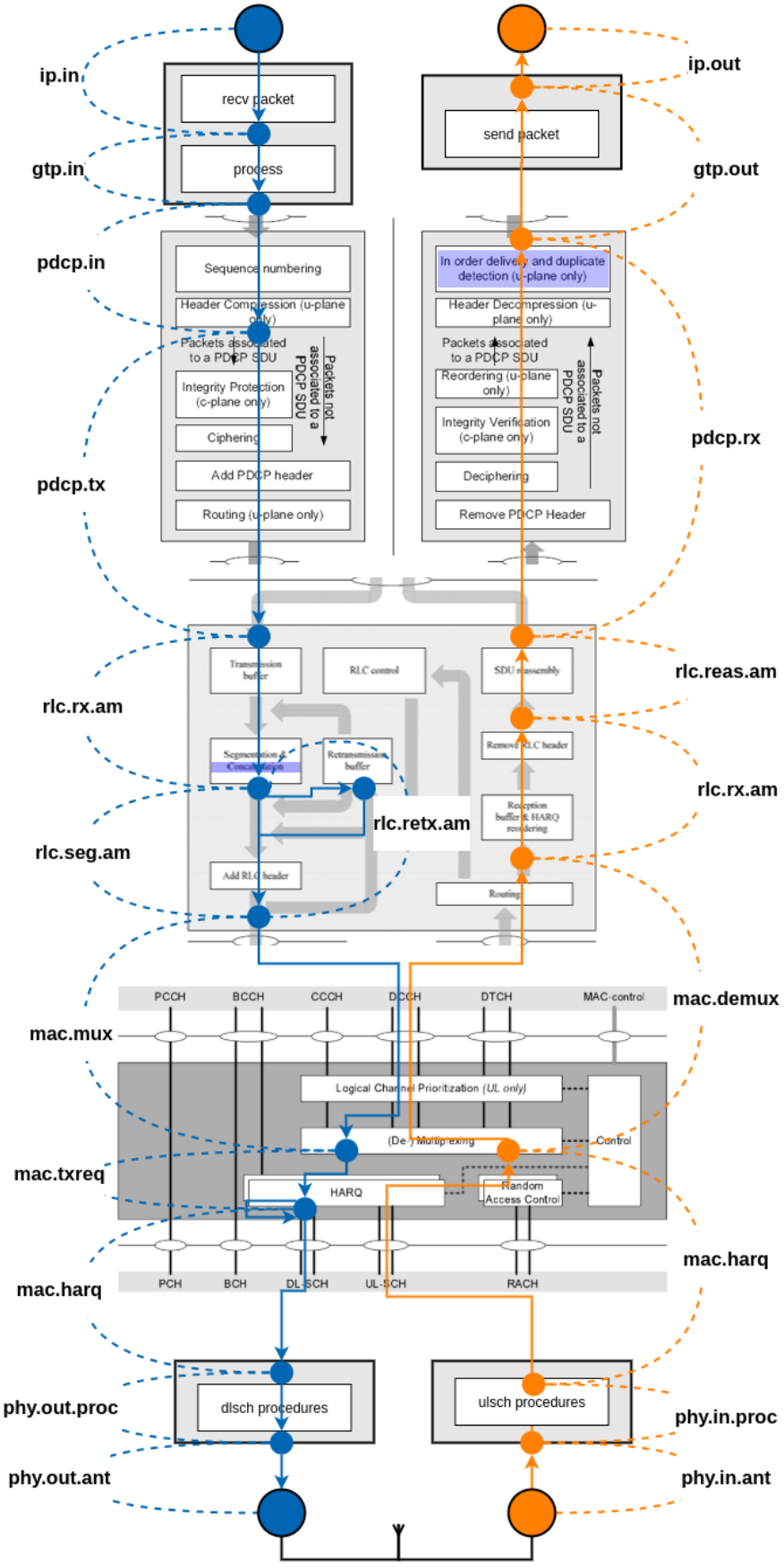
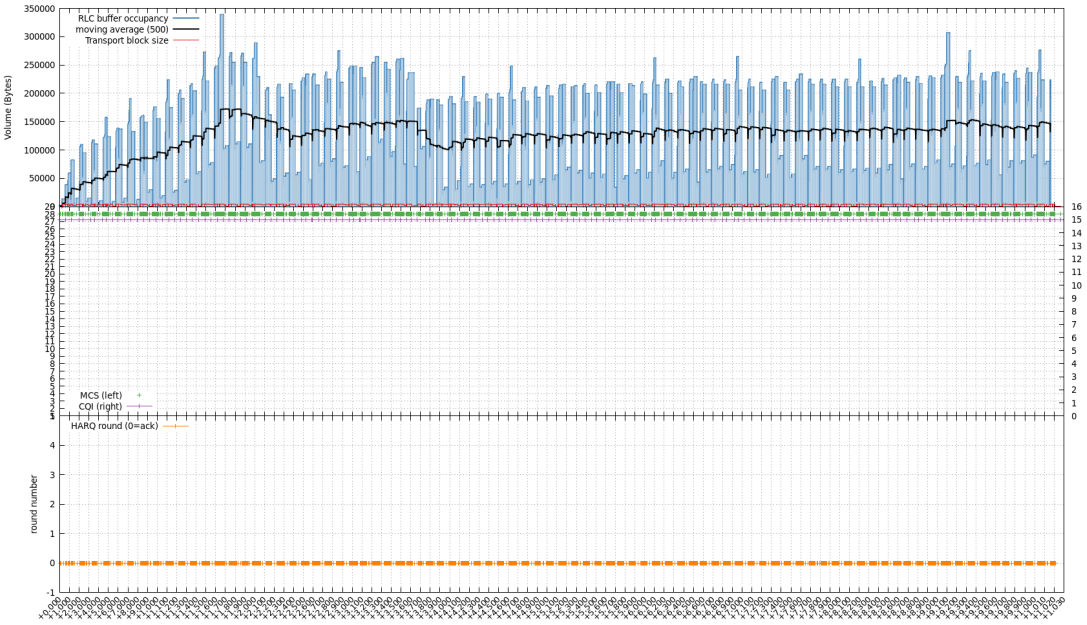



Figure C.4 – LatSeq measurement point in the 3GPP stack.




(a) In the downlink


(b) In the uplink

Figure C.5 – Radio context representation.

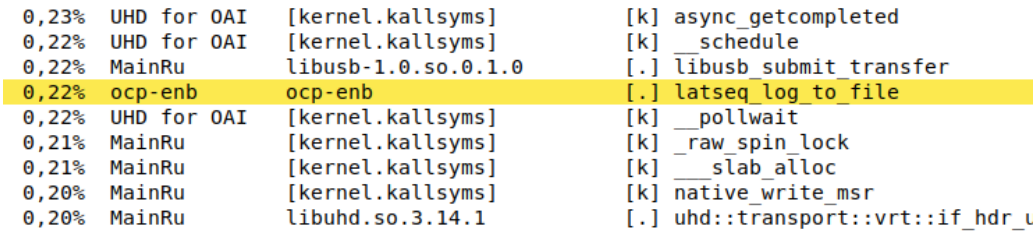



Figure C.6 – CPU usage of LatSeq thread using Linux *prof* tool.

# D Uplink segment as a source of latency and jitter

```
  1 1606323951.068858   172.16.0.4 → 10.193.4.249 GTP <TCP> 110 36114 → 80 [SYN] Seq=0 Win=65535 Len=0
  2 1606323951.069100 10.193.4.249 → 172.16.0.4   GTP <TCP> 112 80 → 36114 [SYN, ACK] Seq=0 Ack=1 Win=
  3 1606323951.090822   172.16.0.4 → 10.193.4.249 GTP <TCP> 102 36114 → 80 [ACK] Seq=1 Ack=1 Win=88064
  4 1606323951.099303   172.16.0.4 → 10.193.4.249 GTP <HTTP> 178 GET / HTTP/1.1
  5 1606323951.099551 10.193.4.249 → 172.16.0.4   GTP <TCP> 104 80 → 36114 [ACK] Seq=1 Ack=77 Win=2905
  6 1606323951.100101 10.193.4.249 → 172.16.0.4   GTP <TCP> 1516 HTTP/1.1 200 OK  [TCP segment of a re
  7 1606323951.100104 10.193.4.249 → 172.16.0.4   GTP <TCP> 1516 80 → 36114 [ACK] Seq=1413 Ack=77 Win=
  8 1606323951.100106 10.193.4.249 → 172.16.0.4   GTP <TCP> 1516 80 → 36114 [ACK] Seq=2825 Ack=77 Win=
  9 1606323951.100108 10.193.4.249 → 172.16.0.4   GTP <TCP> 1516 80 → 36114 [ACK] Seq=4237 Ack=77 Win=
 10 1606323951.100110 10.193.4.249 → 172.16.0.4   GTP <TCP> 1516 80 → 36114 [ACK] Seq=5649 Ack=77 Win=
 11 1606323951.100112 10.193.4.249 → 172.16.0.4   GTP <TCP> 1516 80 → 36114 [ACK] Seq=7061 Ack=77 Win=
 12 1606323951.100113 10.193.4.249 → 172.16.0.4   GTP <TCP> 1516 80 → 36114 [ACK] Seq=8473 Ack=77 Win=
 13 1606323951.100115 10.193.4.249 → 172.16.0.4   GTP <TCP> 1516 80 → 36114 [ACK] Seq=9885 Ack=77 Win=
 14 1606323951.100117 10.193.4.249 → 172.16.0.4   GTP <HTTP> 384 HTTP/1.1 200 OK  (text/html)
 15 1606323951.120821   172.16.0.4 → 10.193.4.249 GTP <TCP> 102 36114 → 80 [ACK] Seq=77 Ack=8473 Win=1
 16 1606323951.129050   172.16.0.4 → 10.193.4.249 GTP <TCP> 102 36114 → 80 [ACK] Seq=77 Ack=11577 Win=
 17 1606323951.150820   172.16.0.4 → 10.193.4.249 GTP <TCP> 102 36114 → 80 [FIN, ACK] Seq=77 Ack=11577
 18 1606323951.151051 10.193.4.249 → 172.16.0.4   GTP <TCP> 104 80 → 36114 [FIN, ACK] Seq=11577 Ack=78
 19 1606323951.170823   172.16.0.4 → 10.193.4.249 GTP <TCP> 102 36114 → 80 [ACK] Seq=78 Ack=11578 Win=
(END)
```

```
1606323951.100149 D pdcp.tx--rlc.tx.am len1466:rnti34029:drb1.psn31.sdu32
1606323951.100149 I rlc.txbuf.am occ8850:rnti34029:drb1
1606323951.100153 D ip.in--gtp.in len1472::ipid42338.teid3396329693.gsn32.gso8.npdu0
1606323951.100154 D gtp.in--pdcp.in.gtp len1464:rnti34029:teid3396329693.drb1.gsn32.gso8.npdu0
1606323951.100154 D pdcp.in--pdcp.tx len1464:rnti34029:drb1.gsn32.psn32
1606323951.100154 D pdcp.tx--rlc.tx.am len1466:rnti34029:drb1.psn32.sdu33
1606323951.100155 I rlc.txbuf.am occ10316:rnti34029:drb1
1606323951.100163 D ip.in--gtp.in len1472::ipid42339.teid3396329693.gsn32.gso8.npdu0
1606323951.100165 D gtp.in--pdcp.in.gtp len1464:rnti34029:teid3396329693.drb1.gsn32.gso8.npdu0
1606323951.100165 D pdcp.in--pdcp.tx len1464:rnti34029:drb1.gsn32.psn33
1606323951.100166 D pdcp.tx--rlc.tx.am len1466:rnti34029:drb1.psn33.sdu34
1606323951.100166 I rlc.txbuf.am occ11782:rnti34029:drb1
1606323951.100172 D ip.in--gtp.in len340::ipid42340.teid3396329693.gsn32.gso8.npdu0
1606323951.100173 D gtp.in--pdcp.in.gtp len332:rnti34029:teid3396329693.drb1.gsn32.gso8.npdu0
1606323951.100173 D pdcp.in--pdcp.tx len332:rnti34029:drb1.gsn32.psn34
1606323951.100173 D pdcp.tx--rlc.tx.am len334:rnti34029:drb1.psn34.sdu35
1606323951.100173 I rlc.txbuf.am occ12116:rnti34029:drb1
1606323951.100717 U phy.in.ant--phy.in.proc len15360::fm122.subfm8
1606323951.100771 D rlc.tx.am--rlc.seg.am len1466:rnti34029:drb1.sdu27.rsn20.rso0
1606323951.100772 D rlc.tx.am--rlc.seg.am len1466:rnti34029:drb1.sdu28.rsn20.rso0
1606323951.100772 D rlc.tx.am--rlc.seg.am len1466:rnti34029:drb1.sdu29.rsn20.rso0
1606323951.100772 D rlc.tx.am--rlc.seg.am len1466:rnti34029:drb1.sdu30.rsn20.rso0
1606323951.100772 D rlc.seg.am--mac.mux len4584:rnti34029:drb1.lcid3.rsn20.rso0.reqfm123
1606323951.100773 I mac.mux mcs28.tbs4587:rnti34029:lcid3
1606323951.100774 D mac.mux--mac.txreq len4587:rnti34029:lcid3.txreq0.reqfm123.harq0.sfn1970
1606323951.100776 D mac.txreq--mac.harq len4587:rnti34029:txreq0.harq0.sfn1970
1606323951.100799 D mac.harq--phy.out.proc len4587:rnti34029:harq0.fm123.subfm2
1606323951.101100 D phy.out.proc--phy.out.ant len15360::fm123.subfm2
1606323951.101717 U phy.in.ant--phy.in.proc len15360::fm122.subfm9
1606323951.101770 D rlc.tx.am--rlc.seg.am len1466:rnti34029:drb1.sdu30.rsn21.rso0
1606323951.101770 D rlc.tx.am--rlc.seg.am len1466:rnti34029:drb1.sdu31.rsn21.rso0
1606323951.101770 D rlc.tx.am--rlc.seg.am len1466:rnti34029:drb1.sdu32.rsn21.rso0
1606323951.101771 D rlc.tx.am--rlc.seg.am len1466:rnti34029:drb1.sdu33.rsn21.rso0
1606323951.101771 D rlc.seg.am--mac.mux len4584:rnti34029:drb1.lcid3.rsn21.rso0.reqfm123
```

Figure D.1 – HTTPS transfer related network captures and LatSeq traces.

The transmission consists in 19 IP packets, 7 in the uplink, and 12 in the downlink for a total transmission volume of 11576 bytes and a transfer duration of 102 ms (see Fig. 4.4 for the components of delays in the completion time). Packets $1-5$ correspond to the connection establishment phase performed in 30.445 ms. The 11576 bytes of data are carried in data packets 6-14 in one burst of 16 ns (instantaneous goodput of 5.8 Gbps). LatSeq traces in Appendix D.1 illustrate fingerprints (highlighted in yellow) associated to HTTP packets 12, 13, 14 in the downlink. The packets of size 1472 bytes at the GTP layer are concatenated into a transport block of size 4584 bytes per TTI. Thus, the data packets

are transmitted at the radio capacity of 36.7 Mbps. The downlink latency of the packet 5 is composed of 0.015 ms for the BS OS network stack, 0.55 ms in the RLC transmission buffer and 0.35 ms for the processing, when the downlink latency for the packet 14 is composed of 0.055 ms for the BS OS stack, 3.4 ms in the RLC transmission buffer and 0.2 ms for the PHY processing. Data packets are acknowledged and the connection is closed with packets 15-19. Acknowledgment packets are transmitted in the uplink in 46 ms, far more than the 6 ms for the data packets in the downlink.

```
/* At the reception of a SR */
UE_info->UE_template[cc_idP][UE_id].ul_SR = 1;
UE_info->UE_template[cc_idP][UE_id].ul_active = TRUE;
```

Listing D.2 – Define the activity of a UE

```
/* Scheduling procedure at the reception of a SR */

// scheduling algorithm
mac->pre_processor_ul.ul(module_idP, CC_id, frameP, subframeP, sched_frame,
    sched_subframeP);

/* ... */
// inside pre_processor_ul.ul(...)

if (B == 0 && UE_to_be_scheduled) {
  // Buffer = 0 and the BS received a SR for this UE
  UE_template->pre_assigned_mcs_ul = 10; // use QPSK mcs only if no CQI
      received
  rb_idx_required[UE_id] = 2; // Allocate RBs (0,1,2)
  continue;
}
/* ... */
// if the user received RBs
if (UE_template_ptr->pre_allocated_nb_rb_ul > 0) {
  UE_template_ptr->ul_SR = 0;
}
```

Listing D.2 – Trigger a grants at the reception of a new scheduling request

```
/* At the reception of a BSR in a TB */

const uint32_t BSR_TABLE[BSR_TABLE_SIZE] = {
  0, 10, 12, ...
};
/* ... */
UE_template_ptr->ul_buffer_info[LCGID0] = BSR_TABLE[bsr];
UE_template_ptr->estimated_ul_buffer =
  UE_template_ptr->ul_buffer_info[LCGID0] +
  UE_template_ptr->ul_buffer_info[LCGID1] +
  UE_template_ptr->ul_buffer_info[LCGID2] +
  UE_template_ptr->ul_buffer_info[LCGID3];
```

Listing D.3 – Update buffer estimation at the reception of a new BSR

```
1  /* At the successful decoding of a data TB */
2  void rx_sdu(...)
3  {
4    /* ... */
5    // Switch between logical channels for all MAC SDU
6    switch (rx_lcid) {
7      /* ... */
8      // User data SDU case
9      case DTCH:
10       /* ... */
11       UE_template_ptr->ul_buffer_info[UE_template_ptr->lcgidmap[rx_lcids[i
             ]]] -= rx_lengths[i];
12   }
13 }
```

Listing D.4 – Update the buffer estimation at the reception of a new transport block

```
1  /* Select UE to be scheduled in the next slot */
2
3  // Decides if the UE is to be scheduled in this slot
4  uint8_t UE_is_to_be_scheduled(...)
5  {
6    if (UE_template->scheduled_ul_bytes < UE_template->estimated_ul_buffer ||
7        UE_template->ul_SR > 0 || // uplink scheduling request
8        (UE_sched_ctl->ul_inactivity_timer > 19 && UE_sched_ctl->ul_scheduled
             == 0) ||  // every 2 frames when RRC_CONNECTED
9        (UE_sched_ctl->ul_inactivity_timer > 10 &&
10       UE_sched_ctl->ul_scheduled == 0 && rrc_status < RRC_CONNECTED) || //
             every Frame when not RRC_CONNECTED
11       (UE_sched_ctl->cqi_req_timer > 300 && rrc_status >= RRC_CONNECTED)) {
12         return 1;
13     }
14 }
15
16 int rr_ul_run(...)
17 {
18   /* ... */
19   const int UE_to_be_scheduled = UE_is_to_be_scheduled(Mod_id, CC_id, UE_id
         , frame, subframe);
```

```
20    const int B = cmax(UE_template->estimated_ul_buffer - UE_template->
          scheduled_ul_bytes, 0);
21    if (B == 0 && !UE_to_be_scheduled)  // UE is not be scheduled, go to the
          next one
22      continue;
23
24    /* if UE has pending scheduling request then pre-allocate 3 RBs */
25    if (B == 0 && UE_to_be_scheduled) {
26      UE_template->pre_assigned_mcs_ul = 10; /* use QPSK mcs only */
27      rb_idx_required[UE_id] = 2;
28      //UE_template->pre_allocated_nb_rb_ul = 3;
29      continue;
30    }
31
32    /* ... */
33
34    // return a pre-allocated resource grid with a number of pre-allocated
          RBs
35    UE_template->pre_first_nb_rb_ul = rbs[r].start;
36    UE_template->pre_allocated_rb_table_index_ul = rb_idx_given[UE_id];
37    UE_template->pre_allocated_nb_rb_ul = rb_table[rb_idx_given[UE_id]];
38  }
```

Code to filter UEs to be scheduled

```
1         hi_dci0_pdu->dci_pdu.dci_pdu_rel8.dci_format = NFAPI_UL_DCI_FORMAT_0;
2         hi_dci0_pdu->dci_pdu.dci_pdu_rel8.rnti = rnti;
3         hi_dci0_pdu->dci_pdu.dci_pdu_rel8.transmission_power = 6000;
4         hi_dci0_pdu->dci_pdu.dci_pdu_rel8.resource_block_start =
              UE_template_ptr->pre_first_nb_rb_ul;
5         hi_dci0_pdu->dci_pdu.dci_pdu_rel8.number_of_resource_block =
6             rb_table[rb_table_index];
7         hi_dci0_pdu->dci_pdu.dci_pdu_rel8.mcs_1 = mcs;
8         hi_dci0_pdu->dci_pdu.dci_pdu_rel8.cyclic_shift_2_for_drms = cshift;
9         hi_dci0_pdu->dci_pdu.dci_pdu_rel8.frequency_hopping_enabled_flag = 0;
10        hi_dci0_pdu->dci_pdu.dci_pdu_rel8.new_data_indication_1 = ndi;
11        hi_dci0_pdu->dci_pdu.dci_pdu_rel8.tpc = tpc;
12        hi_dci0_pdu->dci_pdu.dci_pdu_rel8.cqi_csi_request = cqi_req;
13        hi_dci0_pdu->dci_pdu.dci_pdu_rel8.dl_assignment_index =
14            UE_template_ptr->DAI_ul[sched_subframeP];
15        hi_dci0_pdu->dci_pdu.dci_pdu_rel8.harq_pid = harq_pid;
16        hi_dci0_req_body->sfnsf =
```

```
17              sfnsf_add_subframe(sched_frame, sched_subframeP, 0);
18          hi_dci0_req_body->tl.tag = NFAPI_HI_DCI0_REQUEST_BODY_TAG;
19          hi_dci0_req->sfn_sf = frameP << 4 | subframeP;
20          hi_dci0_req->header.message_id = NFAPI_HI_DCI0_REQUEST;
```

fill DCI0 with grant

# E Enhanced-BSR

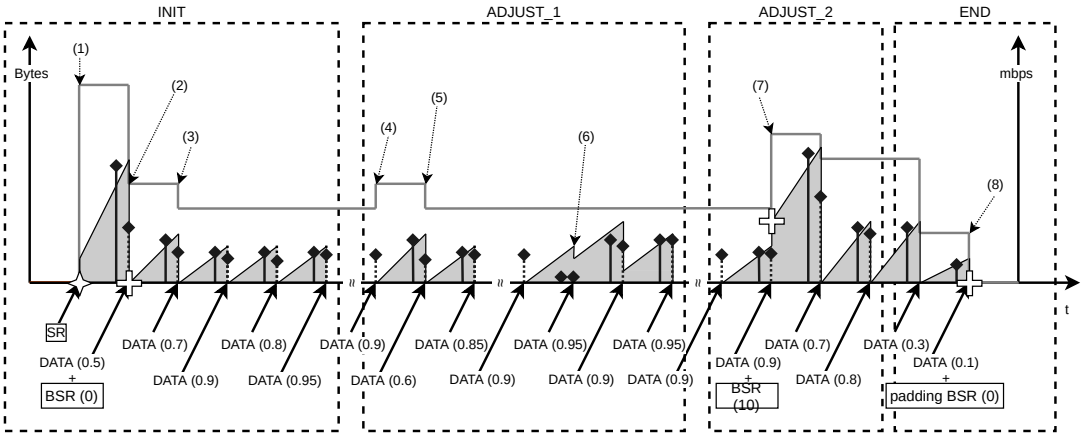



Figure E.1 – Enhanced-BSR estimation dynamic.*Light gray line: estimated datarate; Gray area: estimated buffer size; Star: SR reception; Cross: received BSR value; Pulse line: UG; Dashed pulse line: received TBS.*

The eBSR dynamic illustrated in Appendix E.1 shows the update of the estimated datarate and buffer size according to triggering events.

— (1) Reception of an SR, initialize the estimation model with a value of estimated datarate

— (2) Reception of a transport block with padding, reduction of the estimated datarate (indicated with "DATA (0.5)", 50% of utilization)

— (3) Another reception of a transport block with padding

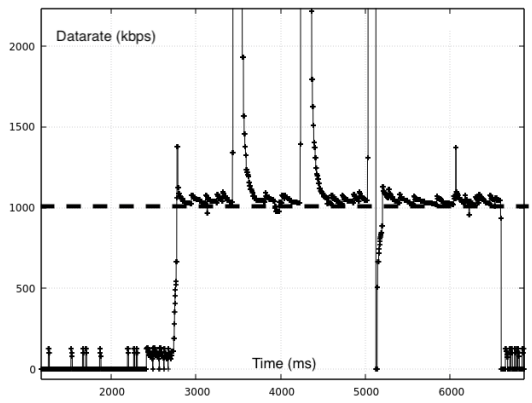
— (4) Probing for extra data when a BSR is not received for a too long time

— (5) Reception of a transport block with padding, rollback to the previous estimated datarate

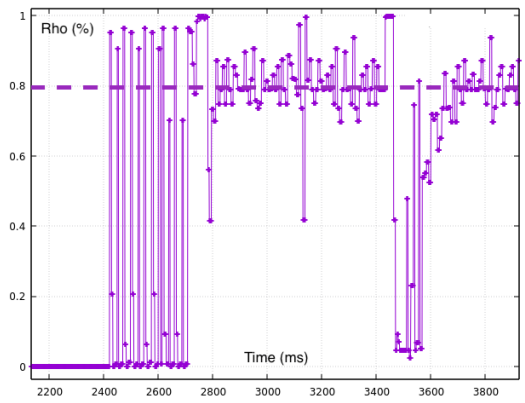
— (6) The UG is not sufficient to empty the estimated buffer size

— (7) At the reception of a non-zero BSR, the estimated buffer size is updated as well as the estimated datarate

— (8) The model detects the end of the transmission after the reception of padding BSR and low TB utilization.




(a) Estimated source datarate $\hat{D}$ (kbps) converges towards $D \rightarrow 1$ Mbps




(b) Transport block utilization $\rho$ (%)

Figure E.2 – Convergence of the model *i.e.* $\hat{D} \rightarrow D$ using the TB utilization $\rho$.

# F Alloc_sim

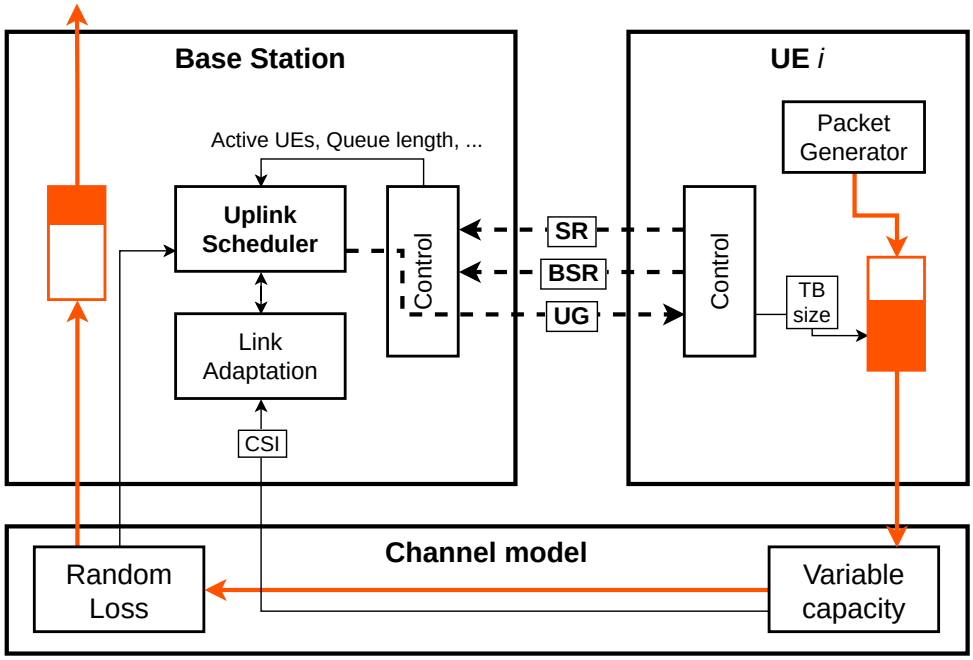



Figure F.1 – Simulation system model for the uplink dynamic scheduling.

## The uplink interface

In the uplink, BS receives packets and acknowledges them. The BS is also in charge of scheduling and issues grants for UEs. The UE generates traffic to be transmitted and send control signalling to request access. The attachment procedure of the UE is not implemented. The UE becomes connected as they send their first SR. The layer 2 of UE and BS, composed of MAC, RLC and PDCP layer in the 3GPP protocol stack is simplified by solely one transmission and one reception queue to store packets. Queues correspond to RLC queues in the 3GPP model. Packet segmentation is enabled when a TBS is not large enough to transmit entirely a data packet.

A great advantage of this simplification is to put apart undefined interactions between resource allocation and complex control plane of 3GPP model. These interactions are assessed in the more realistic OAI testbed.

The limitation it can be found of this model is to not implement control signalling and control mechanism specific to layers, e.g. in-sequence delivery and ARQ at RLC layer or

PDCP ciphering and deciphering, especially, in-sequence delivery could induce important latency and jitter.

## Radio resource allocation schemes

Grant-based access is characterized by the usage of UG to resolve the problem of shared medium between emitting UEs. UE is the exclusive right given to a scheduled UE $i$ to transmit over a RB $k$ on a TTI $t$ with a given Modulation and Coding Schemes. Grants are issued by the Base Station uplink scheduler to indicate radio resources allocated to a user in response to a scheduling request. SR and BSR are the 2 control signaling to request radio resources to MAC scheduler.

The aim of BS MAC scheduler is to assign to each user RBs while respecting a certain number of constraints due to the air interface with an objective of maximizing radio resource usage. We also implemented 2 grant-free methods which are SPS and Fast-UL. Those methods are regularly citing as promising solutions to address 5G latency challenge. As a reminder, the principle of Fast UL is to issue a grant even if the UE does not request it in prevision to a transmission needs. The SPS is the method especially used in Voice over LTE (VoLTE) and deterministic traffics to pre-allocate regularly resources. When SPS is set up, a set of resource block is reserved at a fix period for the UE.

## alloc_sim testbed

The simulation setup is based on *alloc_sim* program (consisting in 2.500 SLOC written in python) which implements a set of UEs connected to a BS through an emulated variable air interface. At every simulation event (a new TTI) the program performs the following procedure:

1. Update air interface capacity
2. Generate new packet arrivals
3. Trigger SR
4. Perform TB transmission and apply losses on TB
5. Decode MAC CE BSR
6. Run scheduling algorithm
7. Perform retransmission

8. Generation the RB map and send UG

After that, new packet arrivals are added to UE transmission buffers. This amount of data is determined either by the configured traffic pattern as a simulation parameter or by the size of the real linux transmission buffer. The configured traffic pattern takes 3 parameters, the packet size, the inter-packet interval and the time range during which the traffic is generated.

When the radio context is updated with data to transmit and the channel properties, UEs send if necessary according to network configurations, scheduling request to the BS. BS collects these scheduling requests for the scheduling stage. In the same time, transmissions from previous grants are performed. Transmissions consist only on removing bytes from UE transmission buffer to add them into the base station reception buffer. As the time in RAN is discretized by slot, the transmission between UE and BS is considered as instantaneous in a TTI slot. At the reception, there is a probability to have HARQ errors that will trigger a retransmission. Along with user data bytes, it is transmitted in the TB, the BSR MAC CE. The code related to the process of SR and BSR signalling is inspired of what is made in OAI (see section 4.2.3).

Sequentially after the effective data transmission, is run the scheduling process to allocate radio resources in the incoming slots. At the core of the radio resource allocation process is the uplink dynamic scheduler. The real value of this work is to ease implementation of scheduling algorithm with standard interfaces in a friendly language for algorithm development (Python). We implemented 4 common best-effort scheduling algorithms, RR, PF, BCQI and Exponential/PF (EXP/PF). The stage of power control is limited to 2 modes, low power and high power according to CQI threshold. The outcomes of the uplink dynamic scheduler is a radio resource blocks map (rbmap) for the slot 8 ms in the future which indicates for each RB, the UE number allowed to transmit on it and the MCS to use (describing the number of bytes carried by the RB).

All along the simulation, packets are observed at their arrival in the transmission buffer (UE) and at their departure from the reception buffer (BS). The output of that is 2 network captures, one gives the packet timestamp at the entry of the RAN and one the timestamp at the output. 1-point capture is used to determine traffic pattern and instantaneous throughput at traffic and user level. The 2 captures are synchronized in time with the common system slot clock. 2-point captures are used to compute time spent by packets in the RAN, *e.g. uplink latency*. Thanks to capture, latency is known at both individual packet level and aggregated cell level, also by flow and by UE.

The energy consumption by UE is calculated at each slot according to the Lauridsen energy model presented in the paragraph 7.3.

## Nfqueue

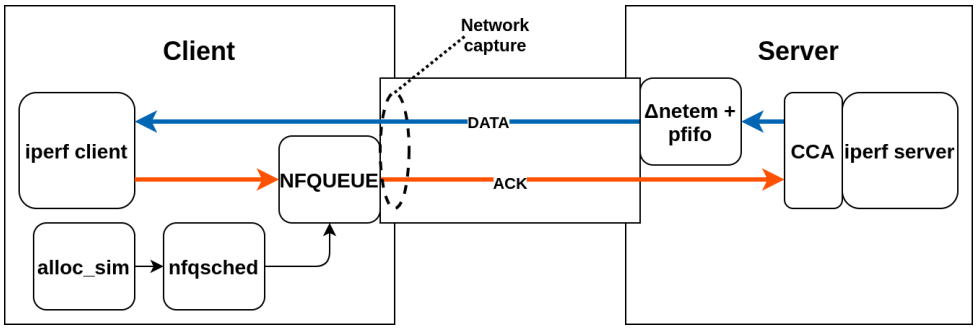



Figure F.2 – *alloc_sim* simulation testbed.

The simulation program can run on itself for uplink allocation simulation experiments. We also design it to be integrated in a linux environment to interact with real IP traffics as presented in Figure F.2. Such emulation mode is interesting to experiment the behaviour of application traffic flow carried by an uplink RAN. To do that, *alloc_sim* is paced to write every millisecond on stdin how many bytes are correctly transmitted by the RAN and to read on stdout the number of bytes awaiting to be transmitted at UE. The `nfqsched` programs make the interface between *alloc_sim* the linux iptable `NFQUEUE`. **nfqueue**[8] for netfilter queue is an iptable target which delegate the decision on packets to a user-space software i.e. `nfqsched`. With the appropriate linux iptable rules, the packets are stored in a buffer waiting to be forwarded to OUTPUT chain. In user-space, the program that uses the `libnetfilter_queue` connect to the NFQUEUE and get the messages from kernel and in return issue verdict on the packet. The emulation reproduces the traffic pattern and specific latency of uplink channel on the real IP traffic. This setup looks like the Direct Code Execution (DCE) of *ns-3* but in a simpler and lighter way or Mahimahi emulation but with a real uplink system emulated.

## Summary

We are aware that the model proposed for this simulation is a bit far from the reality real cellular network. Especially,

8. `https://home.regit.org/netfilter-en/using-nfqueue-and-libnetfilter_queue/`

1. Layer 1 is reduced to a resource grid of block which can carries an amount of bytes according to MCS. In reality, the process to convert bytes to transport block to symbols is in-self, a field of study in latency minimization.
2. The air interface is emulated by 2 values, the CQI and the BLER. In reality the air interface is very much more complex than that with propagation, fading and interference phenomena. We choose to do not implement 3GPP models of propagation and fading loss [259]. We prefer to use channel quality "experienced" by users in a real cell.
3. The 3GPP RRC control plane is totally absent of the simulation. UEs does not have to proceed to RACH procedure to connect to the cellular network or operate handover. It is known that such procedures are source of latency and control overhead but as the simulation aims to focus on radio resource allocation, we put apart control plane not directly related to it.

We designed a simulation program to simplify study work of the uplink radio resource allocation. This program makes easy the development of scheduling algorithm, as well as all mechanism, related to radio resource grid sharing. Ones could argue that the simplification is too important to be representative of real system. Firstly, the simulation testbed is a preliminary stage to experiments realized on OAI testbed. Also, it comes with the advantage of focusing on algorithms design avoiding long development time, by then, accelerating research and development.

# BIBLIOGRAPHY


[1] Sköld Dahlman Parkvall. *5G NR, The next generation wireless access technology*. Elsevier, 2018.

[2] Navid Nikaein et al. « OpenAirInterface: A flexible platform for 5G research ». In: *ACM SIGCOMM Computer Communication Review* 44.5 (2014), pp. 33–38.

[3] Flavien Ronteix–Jacquet et al. « LatSeq: A Low-Impact Internal Latency Measurement Tool for OpenAirInterface ». In: *2021 IEEE Wireless Communications and Networking Conference (WCNC) (IEEE WCNC 2021)*. Nanjing, China, Mar. 2021.

[4] Flavien Ronteix–Jacquet. « Considering return path latency in access networks ». In: *NetSatDay meets NoF: "Fast convergence of congestion control"*. Toulouse (distanciel), France, July 2021. URL: https://hal.archives-ouvertes.fr/hal-03327308.

[5] Flavien Ronteix–Jacquet et al. « Rethinking Buffer Status Estimation to Improve Radio Resource Utilization in Cellular Networks ». In: *IEEE VTC'22 Spring 2021*. Helsinki, Finland, June 2022.

[6] XLR8 consortium. *XLR8: Accelerating Beyond 5G*. Tech. rep. University Carlos III of Madrid, Simula Research Laboratory, Nokia, Orange, LiveU, Software Radio Systems, IMDEA Networks Institute, University of Oslo, Court of the University of Aberdeen, 2016.

[7] Ericsson. *Ericsson Mobility Report 2022*. June 2022. URL: https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/june-2022.

[8] Sandvine. *2022 Global Internet Phenomena Report*. Jan. 2022. URL: https://www.sandvine.com/global-internet-phenomena-report-2022.

[9] E. Dahlman and S. Parkvall. « NR - The New 5G Radio-Access Technology ». In: *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*. June 2018, pp. 1–6. DOI: 10.1109/VTCSpring.2018.8417851.

[10] Jeffrey G Andrews et al. « What will 5G be? » In: *IEEE Journal on selected areas in communications* 32.6 (2014), pp. 1065–1082.

[11] M Series. « IMT Vision–Framework and overall objectives of the future development of IMT for 2020 and beyond ». In: *Recommendation ITU* 2083 (2015). URL: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2083-0-201509-I!!PDF-E.pdf.

[12] Jorge Navarro-Ortiz et al. « A survey on 5G usage scenarios and traffic models ». In: *IEEE Communications Surveys & Tutorials* 22.2 (2020), pp. 905–929.

[13] B. Soret et al. « Fundamental tradeoffs among reliability, latency and throughput in cellular networks ». In: *2014 IEEE Globecom Workshops (GC Wkshps)*. Dec. 2014, pp. 1391–1396. DOI: 10.1109/GLOCOMW.2014.7063628.

[14] 5G Americas. *New services and applications with 5G uRLLC*. Whitepaper. 5G Americas, Oct. 2018. URL: https://www.5gamericas.org/new-services-applications-with-5g-ultra-reliable-low-latency-communications/.

[15] Imtiaz Parvez et al. « A survey on low latency towards 5G: RAN, core network and caching solutions ». In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 3098–3130.

[16] Lennart Schulte et al. « Mobile network delay characteristics and interactions with the transport layer ». PhD thesis. 2017.

[17] Johan Garcia, Stefan Alfredsson, and Anna Brunstrom. « Delay metrics and delay characteristics: A study of four Swedish HSDPA+ and LTE networks ». In: *2015 European Conference on Networks and Communications (EuCNC)*. IEEE. 2015, pp. 234–238.

[18] Haiqing Jiang et al. « Understanding bufferbloat in cellular networks ». In: *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*. 2012, pp. 1–6.

[19] Jim Gettys and Kathleen Nichols. « Bufferbloat: Dark buffers in the internet ». In: *Queue* 9.11 (2011), pp. 40–54.

[20] Toke Høiland-Jørgensen et al. « Measuring latency variation in the internet ». In: *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*. 2016, pp. 473–480.

[21] Luca Schumann et al. « Impact of Evolving Protocols and COVID-19 on Internet Traffic Shares ». In: *arXiv preprint arXiv:2201.00142* (2022). URL: https://arxiv.org/pdf/2201.00142.pdf.

[22] Yihua Guo et al. « Understanding On-Device Bufferbloat for Cellular Upload ». In: *Proceedings of the 2016 Internet Measurement Conference*. IMC '16. Santa Monica, California, USA: Association for Computing Machinery, 2016, pp. 303–317. ISBN: 9781450345262. DOI: 10.1145/2987443.2987490. URL: https://doi.org/10.1145/2987443.2987490.

[23] Stefan Parkvall et al. « 5G NR Release 16: Start of the 5G Evolution ». In: *IEEE Communications Standards Magazine* 4.4 (2020), pp. 56–63.

[24] Xin Fan and Yan Huo. « An Overview of Low latency for Wireless Communications: an Evolutionary Perspective ». In: (2021). arXiv: 2107.03484 [cs.NI]. URL: https://arxiv.org/pdf/2107.03484.pdf.

[25] Samsung. *4G-5G Interworking*. Tech. rep. June 2017. URL: https://images.samsung.com/is/content/samsung/p5/global/business/networks/insights/white-paper/4g-5g-interworking/global-networks-insight-4g-5g-interworking-0.pdf.

[26] Xincheng Zhang. *LTE optimization engineering handbook*. John Wiley & Sons, 2018. ISBN: 9781119158974. DOI: 10.1002/9781119158981. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9781119158981.

[27] Ibrahim Afolabi et al. « Network slicing and softwarization: A survey on principles, enabling technologies, and solutions ». In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), pp. 2429–2453.

[28] Tafseer Akhtar, Christos Tselios, and Ilias Politis. « Radio resource management: approaches and implementations from 4G to 5G and beyond ». In: *Wireless Networks* (2020), pp. 1–42.

[29] 3GPP. *System architecture for the 5G System (5GS)*. Technical Specification (TS) 23.501. 3rd Generation Partnership Project (3GPP), June 2021. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144.

[30] 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description*. Technical Specification (TS) 36.300. 3rd Generation Partnership Project (3GPP), Apr. 2017. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2430.

[31] 3GPP. *NR; Overall description*. Technical Specification (TS) 38.300. 3rd Generation Partnership Project (3GPP), 2018. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specification%20Id=3191.

[32] 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification*. Technical Specification (TS) 36.321. 3rd Generation Partnership Project (3GPP), Apr. 2020. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2437.

[33] 3GPP. *NR; Medium Access Control (MAC) protocol specification*. Technical Specification (TS) 38.321. 3rd Generation Partnership Project (3GPP), Jan. 2018. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3194.

[34] 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA); Packet Data Convergence Protocol (PDCP) specification*. Technical Specification (TS) 36.323. 3rd Generation Partnership Project (3GPP), Aug. 2019. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2439.

[35] 3GPP. *NR; Packet Data Convergence Protocol (PDCP) specification*. Technical Specification (TS) 38.323. 3rd Generation Partnership Project (3GPP), Jan. 2018. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3196.

[36] 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification*. Technical Specification (TS) 36.322. 3rd Generation Partnership Project (3GPP), Aug. 2019. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2438.

[37] 3GPP. *NR; Radio Link Control (RLC) protocol specification*. Technical Specification (TS) 38.321. 3rd Generation Partnership Project (3GPP), Apr. 2019. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3195.

[38] Joseph Ishac and Phil Chimento. *Defining Network Capacity*. RFC 5136. Feb. 2008. DOI: 10.17487/RFC5136. URL: https://www.rfc-editor.org/info/rfc5136.

[39] 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*. Technical Specification (TS) 36.213. 3rd Generation Partnership Project (3GPP), Dec. 2021. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2427.

[40] 3GPP. *5G; NR; Physical layer procedures for data*. Technical Specification (TS) 38.214. 3rd Generation Partnership Project (3GPP), Oct. 2019.

[41] 3GPP. *NR; User Equipment (UE) radio access capabilities*. Technical Specification (TS) 38.306. 3rd Generation Partnership Project (3GPP), 2022. URL: https://www.etsi.org/deliver/etsi_ts/138300_138399/138306/16.07.00_60/ts_138306v160700p.pdf.

[42] Houman Zarrinkoub. *Understanding LTE with MATLAB: from mathematical modeling to simulation and prototyping*. John Wiley & Sons, 2014.

[43] Natale Patriciello et al. « An E2E simulator for 5G NR networks ». In: *Simulation Modelling Practice and Theory* 96 (2019), p. 101933.

[44] Massimo Condoluci et al. « Soft resource reservation for low-delayed teleoperation over mobile networks ». In: *IEEE Access* 5 (2017), pp. 10445–10455.

[45] Dajie Jiang et al. « Principle and performance of semi-persistent scheduling for VoIP in LTE system ». In: *2007 International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE. 2007, pp. 2861–2864.

[46] Jun-Bae Seo and Victor CM Leung. « Performance modeling and stability of semi-persistent scheduling with initial random access in LTE ». In: *IEEE Transactions on Wireless Communications* 11.12 (2012), pp. 4446–4456.

[47] 3GPP. *NR; Physical layer procedures for control*. Technical Specification (TS) 38.213. 3rd Generation Partnership Project (3GPP), Jan. 2018. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3215.

[48] 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding*. Technical Specification (TS) 36.212. 3rd Generation Partnership Project (3GPP), Sept. 2019. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2426.

[49] 3GPP. *NR; Multiplexing and channel coding*. Technical Specification (TS) 38.212. 3rd Generation Partnership Project (3GPP), Aug. 2021. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3214.

[50] K. Pedersen et al. « Agile 5G Scheduler for Improved E2E Performance and Flexibility for Different Network Implementations ». In: *IEEE Communications Magazine* 56.3 (Mar. 2018), pp. 210–217. ISSN: 1558-1896. DOI: 10.1109/MCOM.2017.1700517.

[51] Natale Patriciello et al. « An Improved MAC Layer for the 5G NR Ns-3 Module ». In: *Proceedings of the 2019 Workshop on Ns-3*. WNS3 2019. Florence, Italy: Association for Computing Machinery, 2019, pp. 41–48. ISBN: 9781450371407. DOI: 10.1145/3321349.3321350. URL: https://doi.org/10.1145/3321349.3321350.

[52] Khuram Ashfaq, G. A. Safdar, and M. U. Rehman. « Comparative analysis of scheduling algorithms for radio resource allocation in future communication networks ». In: *PeerJ Computer Science* 7 (2021).

[53] Yinghan Li, Shengqian Han, and Chenyang Yang. « User Scheduling for Uplink OFDMA Systems by Deep Learning ». In: *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2021, pp. 1–5.

[54] Moustafa M Nasralla. « A hybrid downlink scheduling approach for multi-traffic classes in LTE wireless systems ». In: *IEEE Access* 8 (2020), pp. 82173–82186.

[55] Maliheh Mahlouji and Toktam Mahmoodi. « Analysis of Uplink Scheduling for Haptic Communications ». In: *2018 IEEE Globecom Workshops (GC Wkshps)*. IEEE. 2018, pp. 1–7. URL: https://arxiv.org/pdf/1809.09837.pdf.

[56] Francesco Davide Calabrese et al. « Learning radio resource management in RANs: Framework, opportunities, and challenges ». In: *IEEE Communications Magazine* 56.9 (2018), pp. 138–145.

[57] Satheesh Balakrishnan, A. Sivasubramanian, and SPK. « A review of MAC scheduling algorithms in LTE system ». In: *International Journal on Advanced Science, Engineering and Information Technology* 7 (June 2017), p. 1056. DOI: 10.18517/ijaseit.7.3.2059.

[58] Eirini Eleni Tsiropoulou, Aggelos Kapoukakis, and Symeon Papavassiliou. « Uplink resource allocation in SC-FDMA wireless networks: A survey and taxonomy ». In: *Computer Networks* 96 (2016), pp. 1–28.

[59] Najah Abu-Ali et al. « Uplink scheduling in LTE and LTE-advanced: Tutorial, survey and evaluation framework ». In: *IEEE Communications Surveys & Tutorials* 16.3 (2013), pp. 1239–1265.

[60] Hyoungju Ji et al. « Introduction to ultra reliable and low latency communications in 5G ». In: *arXiv preprint arXiv:1704.05565* (2017).

[61] Francesco Davide Calabrese. « Scheduling and Link Adaptation for Uplink SC-FDMA Systems-A LTE Case Study ». PhD thesis. 2009.

[62] Pijush Kanti Dutta Pramanik et al. « Power consumption analysis, measurement, management, and issues: A state-of-the-art review of smartphone battery and energy usage ». In: *IEEE Access* 7 (2019), pp. 182113–182172. URL: https://ieeexplore.ieee.org/ielx7/6287639/8600701/08930492.pdf.

[63] 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*. Technical Specification (TS) 36.123. 3rd Generation Partnership Project (3GPP), July 2021. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2427.

[64] Motorola. *Comparison of UL buffer reporting/scheduling schemes in LTE*. Tech. rep. 2006.

[65] Ericsson. *On the Granularity of Uplink Scheduling in LTE*. Tech. rep. 2006.

[66] KV Pradap, Vinod Ramachandran, and Suresh Kalyanasundaram. « Uplink buffer status reporting for delay constrained flows in 3GPP long term evolution ». In: *2009 IEEE Wireless Communications and Networking Conference*. IEEE. 2009, pp. 1–6.

[67] John Camilo Solano Arenas, Torsten Dudda, and Laetitia Falconetti. « Ultra-low latency in next generation LTE radio access ». In: *SCC 2017; 11th International ITG Conference on Systems, Communications and Coding*. VDE. 2017, pp. 1–6.

[68] Roy T. Fielding, Mark Nottingham, and Julian Reschke. *HTTP Semantics*. RFC 9110. June 2022. DOI: 10.17487/RFC9110. URL: https://www.rfc-editor.org/info/rfc9110.

[69] Eric Rescorla and Tim Dierks. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. Aug. 2008. DOI: 10.17487/RFC5246. URL: https://www.rfc-editor.org/info/rfc5246.

[70] *Internet Protocol*. RFC 791. Sept. 1981. DOI: 10.17487/RFC0791. URL: https://www.rfc-editor.org/info/rfc791.

[71] Jana Iyengar and Martin Thomson. *QUIC: A UDP-Based Multiplexed and Secure Transport*. RFC 9000. May 2021. DOI: 10.17487/RFC9000. URL: https://www.rfc-editor.org/info/rfc9000.

[72] Mike Bishop. *HTTP/3*. RFC 9114. June 2022. DOI: 10.17487/RFC9114. URL: https://www.rfc-editor.org/info/rfc9114.

[73] *Transmission Control Protocol*. RFC 793. Sept. 1981. DOI: 10.17487/RFC0793. URL: https://www.rfc-editor.org/info/rfc793.

[74] Dr. Vern Paxson, Mark Allman, and W. Richard Stevens. *TCP Congestion Control*. RFC 2581. Apr. 1999. DOI: 10.17487/RFC2581. URL: https://www.rfc-editor.org/info/rfc2581.

[75] Van Jacobson. « Congestion avoidance and control ». In: *ACM SIGCOMM computer communication review* 18.4 (1988), pp. 314–329.

[76] Ana Custura, Tom Jones, and Gorry Fairhurst. « Rethinking acks at the transport layer ». In: *2020 IFIP Networking Conference (Networking)*. IEEE. 2020, pp. 731–736.

[77] M. Polese et al. « A Survey on Recent Advances in Transport Layer Protocols ». In: *IEEE Communications Surveys Tutorials* 21.4 (Dec. 2019), pp. 3584–3608. ISSN: 1553-877X. DOI: 10.1109/COMST.2019.2932905.

[78] Ayush Mishra et al. « The Great Internet TCP Congestion Control Census ». In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3.3 (2019), pp. 1–24.

[79] Ayush Mishra, Wee Han Tiu, and Ben Leong. « Are we heading towards a BBR-dominant Internet? » In: (2022). URL: https://www.comp.nus.edu.sg/~bleong/publications/imc2022-nash.pdf.

[80] Keith Winstein and Hari Balakrishnan. « Tcp ex machina: Computer-generated congestion control ». In: *ACM SIGCOMM Computer Communication Review* 43.4 (2013), pp. 123–134.

[81] Josip Lorincz, Zvonimir Klarin, and Julije Ožegović. « A Comprehensive Overview of TCP Congestion Control in 5G Networks: Research Challenges and Future Perspectives ». In: *Sensors* 21.13 (2021), p. 4510.

[82] Reza Poorzare and Anna Calveras. « Challenges on the Way of Implementing TCP over 5G Networks ». In: *IEEE Access* (2020).

[83] Arash Molavi Kakhki et al. « Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols ». In: *Proceedings of the 2017 Internet Measurement Conference*. ACM. 2017, pp. 290–303.

[84] Adam Langley et al. « The quic transport protocol: Design and internet-scale deployment ». In: *Proceedings of the conference of the ACM special interest group on data communication*. 2017, pp. 183–196.

[85] Ke Liu and Jack YB Lee. « On improving TCP performance over mobile data networks ». In: *IEEE transactions on mobile computing* 15.10 (2015), pp. 2522–2536.

[86] Jana Iyengar and Ian Swett. *QUIC Loss Detection and Congestion Control*. RFC 9002. May 2021. DOI: 10.17487/RFC9002. URL: https://rfc-editor.org/rfc/rfc9002.txt.

[87] 3GPP. *Service requirements for the 5G system*. Technical Specification (TS) 22.261. 3rd Generation Partnership Project (3GPP), Mar. 2017. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3107.

[88] Guy Almes et al. *A One-Way Delay Metric for IP Performance Metrics (IPPM)*. RFC 7679. Jan. 2016. DOI: 10.17487/RFC7679. URL: https://rfc-editor.org/rfc/rfc7679.txt.

[89] 3GPP. *5G; Study on Scenarios and Requirements for Next Generation Access Technologies*. Technical Report (TS) 39.913. 3rd Generation Partnership Project (3GPP), May 2017. URL: https://www.etsi.org/deliver/etsi_tr/138900_138999/138913/14.02.00_60/tr_138913v140200p.pdf.

[90] Sunil Kalidindi, Matthew J. Zekauskas, and Dr. Guy T. Almes. *A Round-trip Delay Metric for IPPM*. RFC 2681. Sept. 1999. DOI: 10.17487/RFC2681. URL: https://rfc-editor.org/rfc/rfc2681.txt.

[91] Carlo M. Demichelis and Philip Chimento. *IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)*. RFC 3393. Nov. 2002. DOI: 10.17487/RFC3393. URL: https://rfc-editor.org/rfc/rfc3393.txt.

[92] Ahmad Showail, Kamran Jamshaid, and Basem Shihada. « Buffer sizing in wireless networks: challenges, solutions, and opportunities ». In: *IEEE Communications Magazine* 54.4 (2016), pp. 130–137.

[93] Toke Hoiland-Jorgensen. « Bufferbloat and Beyond - Removing Performance Barriers in Real-World Networks ». MA thesis. Karlstad University, Oct. 2018. URL: http://kau.diva-portal.org/smash/record.jsf?pid=diva2%3A1251705&dswid=3575.

[94] Nicolas Kuhn et al. *Characterization Guidelines for Active Queue Management (AQM)*. RFC 7928. July 2016. DOI: 10.17487/RFC7928. URL: https://rfc-editor.org/rfc/rfc7928.txt.

[95] Stéphane Bortzmeyer. *IETF Recommendations Regarding Active Queue Management*. RFC 7567. RFC Editor, July 2015.

[96] N. Khademi, D. Ros, and M. Welzl. « The new AQM kids on the block: An experimental evaluation of CoDel and PIE ». In: *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. Apr. 2014, pp. 85–90. DOI: 10.1109/INFCOMW.2014.6849173.

[97] R. Adams. « Active Queue Management: A Survey ». In: *IEEE Communications Surveys Tutorials* 15.3 (2013), pp. 1425–1476. DOI: 10.1109/SURV.2012.082212.00018.

[98] Toke Høiland-Jørgensen, Dave Täht, and Jonathan Morton. « Piece of cake: a comprehensive queue management solution for home gateways ». In: *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE. 2018, pp. 37–42.

[99] Miguel Barreiros and Peter Lundqvist. *QoS-Enabled networks: Tools and foundations*. John Wiley & Sons, 2015.

[100] Toke Høiland-Jørgensen et al. *The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm*. RFC 8290. Jan. 2018. DOI: 10.17487/RFC8290. URL: https://rfc-editor.org/rfc/rfc8290.txt.

[101] Rong Pan et al. *Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem*. RFC 8033. Feb. 2017. DOI: 10.17487/RFC8033. URL: https://rfc-editor.org/rfc/rfc8033.txt.

[102] Menglei Zhang et al. « Transport layer performance in 5G mmWave cellular ». In: *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2016, pp. 730–735.

[103] Haiqing Jiang et al. « DRWA: A receiver-centric solution to bufferbloat in cellular networks ». In: *IEEE Transactions on Mobile Computing* 15.11 (2015), pp. 2719–2734.

[104] Menglei Zhang et al. « The bufferbloat problem over intermittent multi-gbps mmwave links ». In: *arXiv preprint arXiv:1611.02117* (2016).

[105] B. Briscoe et al. « Reducing Internet Latency: A Survey of Techniques and Their Merits ». In: *IEEE Communications Surveys Tutorials* 18.3 (July 2016), pp. 2149–2196. ISSN: 1553-877X. DOI: 10.1109/COMST.2014.2375213.

[106] Blesson Varghese et al. « Revisiting the Arguments for Edge Computing Research ». In: *CoRR* abs/2106.12224 (2021). arXiv: 2106.12224. URL: https://arxiv.org/abs/2106.12224.

[107] Fredrik Alriksson et al. « XR and 5G: Extended reality at scale with time-critical communication ». In: *Ericsson technology review* (Aug. 2021). URL: https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/xr-and-5g-extended-reality-at-scale-with-time-critical-communication.

[108] Pavel Mach and Zdenek Becvar. « Mobile edge computing: A survey on architecture and computation offloading ». In: *IEEE Communications Surveys & Tutorials* 19.3 (2017), pp. 1628–1656.

[109] Ke Zhang et al. « Mobile edge computing and networking for green and low-latency Internet of Things ». In: *IEEE Communications Magazine* 56.5 (2018), pp. 39–45.

[110] Jun Wu et al. « Cloud radio access network (C-RAN): a primer ». In: *IEEE Network* 29.1 (2015), pp. 35–41.

[111] Xenofon Foukas et al. « FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks ». In: *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*. CoNEXT '16. Irvine, California, USA: Association for Computing Machinery, 2016, pp. 427–441. ISBN: 9781450342926. DOI: 10.1145/2999572.2999599. URL: https://doi.org/10.1145/2999572.2999599.

[112] Tara Salman. « Cloud RAN: Basics, advances and challenges ». In: *A Surv. C-RAN Basics Virtualization Resour. Alloc. Chall* (2016), pp. 1–16.

[113] Joachim Sachs et al. « Adaptive 5G low-latency communication for tactile internet services ». In: *Proceedings of the IEEE* 107.2 (2018), pp. 325–349.

[114] Kei Sakaguchi et al. « Where, when, and how mmWave is used in 5G and beyond ». In: *IEICE Transactions on Electronics* 100.10 (2017), pp. 790–808.

[115] Amitabha Ghosh et al. « Heterogeneous cellular networks: From theory to practice ». In: *IEEE communications magazine* 50.6 (2012), pp. 54–64.

[116] Chih-Ping Li et al. « 5G ultra-reliable and low-latency systems design ». In: *2017 European Conference on Networks and Communications (EuCNC)*. June 2017, pp. 1–5. DOI: 10.1109/EuCNC.2017.7980747.

[117] Natale Patriciello et al. « 5G New Radio numerologies and their impact on the end-to-end latency ». In: *2018 IEEE 23rd international workshop on computer aided modeling and design of communication links and networks (CAMAD)*. IEEE. 2018, pp. 1–6.

[118] Guillermo Pocovi et al. « MAC layer enhancements for ultra-reliable low-latency communications in cellular networks ». In: *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE. 2017, pp. 1005–1010.

[119] Guillermo Pocovi et al. « On the impact of multi-user traffic dynamics on low latency communications ». In: *2016 International Symposium on Wireless Communication Systems (ISWCS)*. IEEE. 2016, pp. 204–208.

[120] Berker Peköz, Hüseyin Arslan, et al. « Fundamentals of Multi-Numerology 5G New Radio ». In: *arXiv preprint arXiv:1805.02842* (2018).

[121] Gordon J Sutton et al. « Enabling ultra-reliable and low-latency communications through unlicensed spectrum ». In: *IEEE Network* 32.2 (2018), pp. 70–77.

[122] Hyoungju Ji et al. « Ultra-Reliable and Low-Latency Communications in 5G Downlink: Physical Layer Aspects ». In: (2017).

[123] Sofonias Hailu, Mikko Saily, and Olav Tirkkonen. « RRC State handling for 5G ». In: *IEEE Communications Magazine* 57.1 (2018), pp. 106–113.

[124] Nurul Huda Mahmood et al. « Uplink grant-free access solutions for URLLC services in 5G new radio ». In: *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE. 2019, pp. 607–612.

[125] Thilina Nuwan Weerasinghe. « Enabling mMTC and URLLC in 5G: Initial Access, Traffic Prediction, and User Availability ». PhD thesis. 2021.

[126] Hossein Shokri-Ghadikolaei et al. « Millimeter wave cellular networks: A MAC layer perspective ». In: *IEEE Transactions on Communications* 63.10 (2015), pp. 3437–3458.

[127] Icaro Leonardo Da Silva et al. « A novel state model for 5G radio access networks ». In: *2016 IEEE International Conference on Communications Workshops (ICC)*. IEEE. 2016, pp. 632–637.

[128] Natale Patriciello et al. « The Impact of NR scheduling timings on end-to-end delay for uplink traffic ». In: *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2019, pp. 1–6. URL: https://arxiv.org/pdf/1809.09837.pdf.

[129] Zhaowei Tan et al. « Supporting mobile VR in LTE networks: How close are we? » In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2.1 (2018), pp. 1–31.

[130] Brett Levasseur, Mark Claypool, and Robert Kinicki. « Impact of acknowledgments on application performance in 4G LTE networks ». In: *Wireless Personal Communications* 85.4 (2015), pp. 2367–2392.

[131] S. R. Khosravirad et al. « Enhanced HARQ Design for 5G Wide Area Technology ». In: *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*. May 2016, pp. 1–5. DOI: 10.1109/VTCSpring.2016.7504237.

[132] P. Duhamel et al. *Recent advances in HARQ communications*. Research rep. CNRS, 2019.

[133] I. Andriyanova and E. Soljanin. « Optimized IR-HARQ Schemes Based on Punctured LDPC Codes Over the BEC ». In: *IEEE Transactions on Information Theory* 58.10 (Oct. 2012), pp. 6433–6445. ISSN: 1557-9654. DOI: 10.1109/TIT.2012.2203580.

[134] Muhammad Amjad, Leila Musavian, and Mubashir Husain Rehmani. « Effective capacity in wireless networks: A comprehensive survey ». In: *IEEE Communications Surveys & Tutorials* 21.4 (2019), pp. 3007–3038.

[135] Michele Polese et al. « Improved handover through dual connectivity in 5G mmWave mobile networks ». In: *IEEE Journal on Selected Areas in Communications* 35.9 (2017), pp. 2069–2084.

[136] Rajeev Kumar et al. « Dynamic control of RLC buffer size for latency minimization in mobile RAN ». In: *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2018, pp. 1–6.

[137] Thomas Jacobsen et al. « System level analysis of uplink grant-free transmission for URLLC ». In: *2017 IEEE Globecom Workshops (GC Wkshps)*. IEEE. 2017, pp. 1–6.

[138] 3GPP. *Policy and charging control framework for the 5G System (5GS)*. Technical Specification (TS) 23.503. 3rd Generation Partnership Project (3GPP), Dec. 2017. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3334.

[139] Ericsson et al. *Use of L4S in 5GS*. Tech. rep. May 2021.

[140] Davide Brunello and Mustafa Ozger. « L4S in 5G networks ». MA thesis. KTH Royal Institute of Technology, 2020. URL: https://kth.diva-portal.org/smash/record.jsf?aq2=%5B%5B%5D%5D&aq2=%5B%5B%5D%5D&c=1&c=1&af=%5B%5D&af=%5B%5D&searchType=SIMPLE&searchType=SIMPLE&sortOrder2=title_sort_asc&sortOrder2=title_sort_asc&query=brunello&query=brunello&language=en&language=en&pid=diva2%3A1484466&pid=diva2%3A1484466&aq=%5B%5B%5D%5D&aq=%5B%5B%5D%5D&sf=all%3E&sf=all&aqe=%5B%5D&aqe=%5B%5D&sortOrder=author_sort_asc&sortOrder=author_sort_asc&onlyFullText=false&onlyFullText=false&noOfRows=50&noOfRows=50&dswid=9588.

[141] Ed. G. White CableLabs K. De Schepper Nokia Bell Labs B. Briscoe. *DualQ Coupled AQMs for Low Latency, Low Loss and Scalable Throughput (L4S)*. RFC draft-ietf-tsvwg-aqm-dualq-coupled-10. RFC Editor, July 2019.

[142] J. D. Beshay et al. « On active queue management in cellular networks ». In: *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. May 2017, pp. 384–389. DOI: 10.1109/INFCOMW.2017.8116407.

[143] Mikel Irazabal et al. « Preventing RLC Buffer Sojourn Delays in 5G ». In: *IEEE Access* 9 (2021), pp. 39466–39488.

[144] Soheil Abbasloo and H Jonathan Chao. « Bounding queue delay in cellular networks to support ultra-low latency applications ». In: *arXiv preprint arXiv:1908.00953* (2019).

[145] Mikel Irazabal, Elena Lopez-Aguilera, and Ilker Demirkol. « Active queue management as quality of service enabler for 5G networks ». In: *2019 European Conference on Networks and Communications (EuCNC)*. IEEE. 2019, pp. 421–426.

[146] Y. Dai et al. « Channel quality aware active queue management in cellular networks ». In: *2017 9th Computer Science and Electronic Engineering (CEEC)*. Sept. 2017, pp. 183–188. DOI: 10.1109/CEEC.2017.8101622.

[147] Y. Dai et al. « Channel quality aware active queue management in cellular networks ». In: *2017 9th Computer Science and Electronic Engineering (CEEC)*. Sept. 2017, pp. 183–188. DOI: 10.1109/CEEC.2017.8101622.

[148] Joseph D Beshay et al. « On active queue management in cellular networks ». In: *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2017, pp. 384–389.

[149] Yifeng Tan. « Active queue management for LTE uplink in eNodeB ». en. G2 Pro gradu, diplomityö. Aalto university, Espoo, 2009, p. 83. URL: http://urn.fi/URN:NBN:fi:aalto-201203071276.

[150] Navid Ehsan and Thomas Klingenbrunn. *Delay based active queue management for uplink traffic in user equipment*. July 2016.

[151] Riikka Susitaival, Yifeng Tan, and Per Johan Torsner. *Active queue management for wireless communication network uplink*. Feb. 2015.

[152] Marco Mezzavilla et al. « End-to-end simulation of 5G mmWave networks ». In: *IEEE Communications Surveys & Tutorials* 20.3 (2018), pp. 2237–2263.

[153] Jung Hyun Bae et al. « An overview of channel coding for 5G NR cellular communications ». In: *APSIPA Transactions on Signal and Information Processing* 8 (2019), e17. DOI: 10.1017/ATSIP.2019.10.

[154] Jeon. « NR Wide Bandwidth Operations ». In: *IEEE Communications* (2017).

[155] Alexios Balatsoukas-Stimming, Pascal Giard, and Andreas Burg. « Comparison of polar decoders with existing low-density parity-check and turbo decoders ». In: *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. Ieee. 2017, pp. 1–6.

[156] Ghosh. *5G NR: Physical Layer overview and performances*. Tech. rep. Nokia Bell Labs, 2018. URL: https://ctw2018.ieee-ctw.org/files/2018/05/5G-NR-CTW-final.pdf.

[157] Masterson. *Massive MIMO and Beamforming: The Signal Processing Behind the 5G Buzzwords*. Tech. rep. analog Devices, 2017.

[158] Xiang Gao et al. « Massive MIMO in real propagation environments ». In: *IEEE Trans. Wireless Commun* (2014).

[159] Ana Custura, Tom Jones, and Gorry Fairhurst. « Impact of Acknowledgements using IETF QUIC on Satellite Performance ». In: *2020 10th Advanced Satellite Multimedia Systems Conference and the 16th Signal Processing for Space Communications Workshop (ASMS/SPSC)*. IEEE. 2020, pp. 1–8.

[160] Tong Li et al. « Tack: Improving wireless transport performance by taming acknowledgments ». In: *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 2020, pp. 15–30.

[161] Mohit P Tahiliani, Vishal Misra, and KK Ramakrishnan. « A principled look at the utility of feedback in congestion control ». In: *Proceedings of the 2019 Workshop on Buffer Sizing*. 2019, pp. 1–5.

[162] Feng Lu et al. « CQIC: Revisiting cross-layer congestion control for cellular networks ». In: *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. 2015, pp. 45–50.

[163] A Jain et al. « Mobile throughput guidance inband signaling protocol ». In: *IETF, work in progress* (2015), pp. 1–16.

[164] Mauro Cociglio et al. *Explicit Flow Measurements Techniques*. Internet-Draft draft-mdt-ippm-explicit-flow-measurements-01. Internet Engineering Task Force, Feb. 2021. 39 pp. URL: https://datatracker.ietf.org/doc/html/draft-mdt-ippm-explicit-flow-measurements-01.

[165] Ingemar Johansson. *Motivation to improved ECN handling in NR*. Technical Report R2-1709469. 3GPP TSG-RAN WG2, 2017. URL: http://www.3gpp.org/ftp/tsg_ran/WG2_RL2/TSGR2_99/Docs/R2-1709469.zip.

[166] Lingfeng Guo and Jack YB Lee. « Stateful-TCP—A New Approach to Accelerate TCP Slow-Start ». In: *IEEE Access* 8 (2020), pp. 195955–195970.

[167] Prateesh Goyal, Mohammad Alizadeh, and Hari Balakrishnan. « Rethinking congestion control for cellular networks ». In: *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. 2017, pp. 29–35.

[168] Philipp Bruhn. « Performance and Improvements of TCP CUBIC in Low-Delay Cellular Networks ». In: ().

[169] Ahmed Nasrallah et al. « Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research ». In: *IEEE Communications Surveys & Tutorials* 21.1 (2018), pp. 88–145.

[170] 5G-PPP. *View on 5G Architecture*. Tech. rep. Architecture Working Group, June 2019.

[171] Bruno Astuto A Nunes et al. « A survey of software-defined networking: Past, present, and future of programmable networks ». In: *IEEE Communications surveys & tutorials* 16.3 (2014), pp. 1617–1634.

[172] 3GPP. *Study on latency reduction techniques for LTE*. Technical Report (TR) 36.8811. 3rd Generation Partnership Project (3GPP), July 2016. URL: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2901.

[173] Davide Brunello et al. « Low Latency Low Loss Scalable Throughput in 5G Networks ». In: *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. 2021, pp. 1–7. DOI: 10.1109/VTC2021-Spring51267.2021.9448764.

[174] Magnus Austrheim. « Implementing immediate forwarding for 4G in a network simulator ». PhD thesis. Oslo University, 2018.

[175] Muhammad Imran, Abas Md Said, and Halabi Hasbullah. « A survey of simulators, emulators and testbeds for wireless sensor networks ». In: *2010 International Symposium on Information Technology*. Vol. 2. IEEE. 2010, pp. 897–902.

[176] OSA. *OpenAirInteface 5G website*.

[177] Mla Vilakazi et al. « Evaluating an Evolving OAI Testbed: Overview of Options, Building tips, and Current Performance ». In: *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*. Vol. 1. IEEE. 2021, pp. 818–827. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9442028.

[178] Ismael Gomez-Miguelez et al. « SrsLTE: An Open-Source Platform for LTE Evolution and Experimentation ». In: *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*. WiNTECH '16. New York City, New York: Association for Computing Machinery, 2016, pp. 25–32. ISBN: 9781450342520. DOI: 10.1145/2980159.2980163. URL: https://doi.org/10.1145/2980159.2980163.

[179] Aymeric de Javel et al. « Towards a new open-source 5G development framework: an introduction to free5GRAN ». In: *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. IEEE. 2021, pp. 1–5.

[180] Ajay Tirumala. « Iperf: The TCP/UDP bandwidth measurement tool ». In: *http://dast. nlanr. net/Projects/Iperf/* (1999).

[181] litespeed. *lsquic*.

[182] Toke Høiland-Jørgensen et al. « Flent: The flexible network tester ». In: *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*. 2017, pp. 120–125.

[183] K. Hedayat et al. *A Two-Way Active Measurement Protocol (TWAMP)*. RFC 5357. Oct. 2008. DOI: 10.17487/RFC5357. URL: https://www.rfc-editor.org/info/rfc5357.

[184] Al Morton. *Active and Passive Metrics and Methods (with Hybrid Types In-Between)*. RFC 7799. May 2016. DOI: 10.17487/RFC7799. URL: https://www.rfc-editor.org/info/rfc7799.

[185] Mark Allman. « Comments on bufferbloat ». In: *ACM SIGCOMM Computer Communication Review* 43.1 (2012), pp. 30–37.

[186] Amaury Van Bemten and Wolfgang Kellerer. « Network calculus: A comprehensive guide ». In: (2016).

[187] Krishnaiyan Thulasiraman and Madisetti NS Swamy. *Graphs: theory and algorithms*. John Wiley & Sons, 2011.

[188] Yuanjie Li et al. « Mobileinsight: Extracting and analyzing cellular network information on smartphones ». In: *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. 2016, pp. 202–215.

[189] IBM. *Logging in multi-threaded applications efficiently with ring buffer*. 2007. URL: https://www.ibm.com/developerworks/aix/library/au-buffer/index.html.

[190] Apache. *Log4j*. 2014. URL: https://logging.apache.org/log4j/2.x/misc/async.html.

[191] H. Simpson. *zlog*. URL: https://github.com/HardySimpson/zlog.

[192] A. Afanasyev. *Optimize multi-threaded logging using lock-free queue and separate thread*. 2016. URL: https://redmine.named-data.net/issues/2513.

[193] *The bufferbloat project*. URL: https://www.bufferbloat.net/projects/.

[194] Chiara Chirichella and Dario Rossi. « To the Moon and back: are Internet bufferbloat delays really that large? » In: *2013 Proceedings IEEE INFOCOM*. IEEE. 2013, pp. 3297–3302.

[195] Natalie Larson et al. « Investigating Excessive Delays in Mobile Broadband Networks ». In: *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. AllThingsCellular '15. London, United Kingdom: Association for Computing Machinery, 2015, pp. 51–56. ISBN: 9781450335386. DOI: 10.1145/2785971.2785980. URL: https://doi.org/10.1145/2785971.2785980.

[196] Markus Laner et al. « A comparison between one-way delays in operating HSPA and LTE networks ». In: *2012 10th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*. IEEE. 2012, pp. 286–292.

[197] Megha Sahu, Snigdha Damle, and Arzad Alam Kherani. « End-to-end uplink delay jitter in LTE systems ». In: *Wireless Networks* 27.3 (2021), pp. 1783–1800.

[198] Nadhif Muhammad. *Iperf, Ping, and slicing in RAN testing*.

[199] Mads Lauridsen. *Studies on mobile terminal energy consumption for LTE and future 5G*. Department of Electronic Systems, Aalborg University, 2015. URL: https://core.ac.uk/download/pdf/60598258.pdf.

[200] Guillermo Pocovi et al. « On the suitability of LTE air interface for reliable low-latency applications ». In: *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2019, pp. 1–6.

[201] OSA. *OpenAirInteface 5G source code*.

[202] Gomez et al. *srsRAN source code for BSR management*. URL: %5Curl%7Bhttps://github.com/srsran/srsRAN/blob/master/srsenb/src/stack/mac/common/ue_buffer_manager.cc%7D.

[203] Shinik Park et al. « Exll: an extremely low-latency congestion control for mobile cellular networks ». In: *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*. 2018, pp. 307–319.

[204] Nikaein et al. *OAI source code for MAC layer*. URL: %5Curl%7Bhttps://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/master/openair2/LAYER2/MAC%7D.

[205] Thomas Pötsch. *Future Mobile Transport Protocols*. Springer, 2016.

[206] Megha Sahu and Arzad A Kherani. « End-to-End Delay Jitter in LTE Uplink: Simple Models, Empirical Validation & Applications ». In: *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE. 2020, pp. 221–228.

[207] Philipp Bruhn, Mirja Kuehlewind, and Maciej Muehleisen. « Performance and Improvements of TCP CUBIC in Low-Delay Cellular Networks ». In: *2022 IFIP Networking Conference (IFIP Networking)*. IEEE. 2022, pp. 1–9.

[208] Ingemar Johansson. *Congestion Control for 4G/5G Networks IETF 96*. RFC. RFC Editor, July 2016. URL: https://datatracker.ietf.org/meeting/96/materials/slides-96-iccrg-1.

[209] Ingemar Johansson. *Congestion control for 4G and 5G access*. Internet-Draft draft-johansson-cc-for-4g-5g-02. Internet Engineering Task Force, July 2016. 14 pp. URL: https://datatracker.ietf.org/doc/html/draft-johansson-cc-for-4g-5g-02.

[210] Remi Robert et al. « Behaviour of common TCP variants over LTE ». In: *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2016, pp. 1–7.

[211] Srivastava Ashutosh, Fraida Fund, and Shivendra S Panwar. « An experimental evaluation of low latency congestion control for mmwave links ». In: *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2020, pp. 352–357. URL: http://witestlab.poly.edu/~ffund/pubs/tcp-mmwave.pdf.

[212] Junxian Huang et al. « An in-depth study of LTE: effect of network protocol and application behavior on performance ». In: *ACM SIGCOMM Computer Communication Review* 43.4 (2013), pp. 363–374.

[213] Marcus Pieska and Andreas Kassler. « TCP performance over 5G mmWave links—Tradeoff between capacity and latency ». In: *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE. 2017, pp. 385–394.

[214] David L. Black. *Relaxing Restrictions on Explicit Congestion Notification (ECN) Experimentation*. RFC 8311. Jan. 2018. DOI: 10.17487/RFC8311. URL: https://www.rfc-editor.org/info/rfc8311.

[215] M. Allman, V. Paxson, and E. Blanton. *TCP Congestion Control*. RFC 5681. RFC Editor, Sept. 2009. URL: http://www.rfc-editor.org/rfc/rfc5681.txt.

[216] Neal Cardwell et al. « BBR: Congestion-based congestion control ». In: *Queue* 14.5 (2016), pp. 20–53.

[217] Leonard Kleinrock. « Internet congestion control using the power metric: Keep the pipe just full, but no fuller ». In: *Ad Hoc Networks* 80 (2018), pp. 142–157. ISSN: 1570-8705. DOI: 10.1016/j.adhoc.2018.05.015. URL: https://www.sciencedirect.com/science/article/pii/S1570870518302476.

[218] Andrei Gurtov. « Effect of delays on TCP performance ». In: *Emerging Personal Wireless Communications*. Springer, 2002, pp. 87–105.

[219] Somaya Arianfar et al. « Proceedings of Seminar on Network Protocols in Operating Systems ». In: (2013). URL: %7Bhttps://aaltodoc.aalto.fi/bitstream/handle/123456789/8945/isbn9789526049977.pdf%7D.

[220] Michael Scharf, Marc Necker, and Bernd Gloss. « The sensitivity of TCP to sudden delay variations in mobile networks ». In: *International Conference on Research in Networking*. Springer. 2004, pp. 76–87.

[221] Matt Sargent et al. *Computing TCP's Retransmission Timer*. RFC 6298. June 2011. DOI: 10.17487/RFC6298. URL: https://www.rfc-editor.org/info/rfc6298.

[222] Eneko Atxutegi et al. « On the use of TCP BBR in cellular networks ». In: *IEEE Communications Magazine* 56.3 (2018), pp. 172–179.

[223] Rajeev Kumar et al. « TCP BBR for ultra-low latency networking: challenges, analysis, and solutions ». In: *2019 IFIP Networking Conference (IFIP Networking)*. IEEE. 2019, pp. 1–9.

[224] Jana Iyengar and Ian Swett. *QUIC Acknowledgement Frequency*. Internet-Draft draft-ietf-quic-ack-frequency-00. Internet Engineering Task Force, July 2021. 10 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-quic-ack-frequency-00.

[225] Gorry Fairhurst. *Current QUIC performance is compromised compared to TCP for asymmetric paths*. Apr. 2020. URL: https://mailarchive.ietf.org/arch/msg/quic/NZBbuY88v0lrVspQE2A5br83asA/.

[226] Sangtae Ha and Injong Rhee. « Hybrid slow start for high-bandwidth and long-distance networks ». In: *Proc. PFLDnet*. 2008, pp. 1–6.

[227] Benjamin Peters et al. « TCP HyStart Performance over a Satellite Network ». In: *Proceedings of the 0x15 NetDev Conference*. 2021.

[228] Philipp Bruhn and Maciej Muehleisen. « Behavior of TCP CUBIC in Low-Latency Mobile Radio Networks ». In: (Aug. 2020). URL: https://datatracker.ietf.org/meeting/interim-2020-maprg-01/materials/slides-interim-2020-maprg-01-sessa-behavior-of-tcp-cubic-in-low-latency-mobile-radio-networks-00.pdf.

[229] Tomofumi Koyama and Katsunori Aoki. « Slow start algorithm for mobile broadband networks including delay unrelated to network congestion ». In: *2015 International Conference on Computing, Networking and Communications (ICNC)*. IEEE. 2015, pp. 148–152.

[230] Neil Agarwal et al. « Mind the delay: the adverse effects of delay-based TCP on HTTP ». In: *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*. 2020, pp. 364–370.

[231] Johan Garcia, Stefan Alfredsson, and Anna Brunstrom. « Examining TCP short flow performance in cellular networks through active and passive measurements ». In: *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. 2015, pp. 7–12.

[232] Ilpo Järvinen et al. « Effect of competing TCP traffic on interactive real-time communication ». In: *International Conference on Passive and Active Network Measurement*. Springer. 2013, pp. 94–103.

[233] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. « Stochastic forecasts achieve high throughput and low delay over cellular networks ». In: *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*. 2013, pp. 459–471.

[234] Michael Welzl et al. « Future Internet Congestion Control: The Diminishing Feedback Problem ». In: *IEEE Communications Magazine* (2022).

[235] Yasir Zaki et al. « Adaptive congestion control for unpredictable cellular networks ». In: *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 2015, pp. 509–522.

[236] Mingyan Li et al. « Predictive pre-allocation for low-latency uplink access in industrial wireless networks ». In: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE. 2018, pp. 306–314.

[237] Peter Moberg et al. *Prohibiting unnecessary scheduling requests for uplink grants*. Nov. 2013.

[238] Björn Nordström, Eddie Corbett, and Ying Sun. *Scheduling of delay-sensitive traffic*. Aug. 2015.

[239] Ye Feng, Ampalavanapillai Nirmalathas, and Elaine Wong. « A Predictive Semi-Persistent Scheduling Scheme for Low-Latency Applications in LTE and NR Networks ». In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE. 2019, pp. 1–6.

[240] Par Ankel et al. *Uplink prescheduling*. Nov. 2016.

[241] Aby K Abraham. « An optimized method of buffer status reporting for uplink data in lte ». In: *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE. 2015, pp. 1–4.

[242] Qi Zhang, Alexandros Nikou, and Marios Daoutis. « Predicting Buffer Status Report (BSR) for 6G Scheduling using Machine Learning Models ». In: *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2022, pp. 632–637.

[243] Neung-Hyung Lee and Sungoh Kwon. « Uplink QoS scheduling algorithm with delay estimation for LTE systems ». In: *Wireless Personal Communications* 80.3 (2015), pp. 1131–1146.

[244] Yiu Bun Jack Lee and Chi Fung Chan. *Method for link buffer size and queue length estimation for bandwidth-varying mobile data networks*. Dec. 2014. URL: https://patentimages.storage.googleapis.com/6b/67/3f/0c8f416cef1a04/US8923270.pdf.

[245] Amr Rizk and Markus Fidler. « Queue-aware uplink scheduling with stochastic guarantees ». In: *Computer Communications* 84 (2016), pp. 63–72.

[246] Vikram Chandrasekhar et al. *Timer-based scheme for user equipment queue state estimation*. Sept. 2017.

[247] Eneko Atxutegi et al. « TCP Performance over Current Cellular Access: A Comprehensive Analysis ». In: *Autonomous Control for a Reliable Internet of Services*. Springer, Cham, 2018, pp. 371–400.

[248] Cédric Gueguen and Malo Manini. « Dynamic tradeoff between energy and throughput in wireless 5G networks ». In: *Wireless communications and mobile computing* 2018 (2018).

[249] F. Capozzi et al. « Downlink Packet Scheduling in LTE Cellular Networks: Key Design Issues and a Survey ». In: *IEEE Communications Surveys Tutorials* 15.2 (2013), pp. 678–700. ISSN: 2373-745X. DOI: 10.1109/SURV.2012.060912.00100.

[250] Obada Al-Khatib, Wibowo Hardjawana, and Branka Vucetic. « Channel-and buffer-aware scheduling and resource allocation algorithm for LTE-A uplink ». In: *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*. IEEE. 2014, pp. 1001–1006.

[251] Giuseppe Piro et al. « Two-level downlink scheduling for real-time multimedia services in LTE networks ». In: *IEEE Transactions on Multimedia* 13.5 (2011), pp. 1052–1065.

[252] Mads Lauridsen, Preben Mogensen, and Laurent Noël. « Empirical LTE smartphone power model with DRX operation for system level simulations ». In: *2013 IEEE 78th vehicular technology conference (VTC Fall)*. IEEE. 2013, pp. 1–6. URL: https://vbn.aau.dk/ws/files/194890384/UEpowerModel2013.pdf.

[253] Jana Iyengar and Ian Swett. *QUIC Acknowledgement Frequency*. Internet-Draft draft-ietf-quic-ack-frequency-02. Internet Engineering Task Force, July 2022. 13 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-quic-ack-frequency/02/.

[254] Mingyan Li et al. « A Learning-Based Pre-Allocation Scheme for Low-Latency Access in Industrial Wireless Networks ». In: *IEEE Transactions on Wireless Communications* 19 (2020), pp. 650–664.

[255] Yuanxin Sun et al. « Latency Performance Analysis of Predictive Resource Allocation ». In: *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)* (2019), pp. 1–6.

[256] Samad Ali, Nandana Rajatheva, and Walid Saad. « Fast uplink grant for machine type communications: Challenges and opportunities ». In: *IEEE Communications Magazine* 57.3 (2019), pp. 97–103. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8663999.

[257] Nusrat Afrin, Jason Brown, and Jamil Y Khan. « An adaptive buffer based semi-persistent scheduling scheme for machine-to-machine communications over LTE ». In: *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*. IEEE. 2014, pp. 260–265. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6982926.

[258] Lianming Zhang et al. « LNTP: An end-to-end online prediction model for network traffic ». In: *IEEE Network* 35.1 (2020), pp. 226–233.

[259] 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception*. Technical Specification (TS) 36.101. 3rd Generation Partnership Project (3GPP), Jan. 2021. URL: https://portal.

3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2411.

# List of Figures


0.1 Emergence of 5G in mobile habits and new services . . . 2
0.2 Thesis chapters outline . . . 6

1.1 LTE and 5G networks architecture with a focus on the user-plane . . . 9
1.2 LTE and 5G protocol stack . . . 12
1.3 Uplink radio resource grid . . . 15
1.4 Uplink radio resource allocation schemes . . . 18
1.5 Uplink grant-based dynamic scheduling . . . 20
1.6 3GPP Buffer Status Reports . . . 23
1.7 ACKs for TCP congestion control . . . 27
1.8 End-to-End latency segments . . . 29
1.9 CDF of latency and definitions . . . 32
1.10 Packet delay exhibiting bufferbloat . . . 33

2.1 Degree of realism of the different research methods . . . 44
2.2 Mobile network core architecture . . . 48
2.3 Photos of the OAI RAN testbed . . . 50
2.4 Capture points on the OAI RAN testbed . . . 52
2.5 Transport protocols and applications . . . 53
2.6 2-points passive network capture . . . 56
2.7 TCP time-sequence plot reading . . . 57

3.1 Internal latency cases . . . 64
3.2 LatSeq fingerprint . . . 66
3.3 LatSeq journeys . . . 68
3.4 Matrix of internal latency . . . 69
3.5 LatSeq architecture . . . 70
3.6 LatSeq fingerprint data structures . . . 73
3.7 Screen capture of the LatSeq fingerprints file . . . 75
3.8 LatSeq points in the RAN testbed . . . 76

3.9 ECDF of latency from journey . . . 83
3.10 Waterfall representation of packet journeys . . . 84
3.11 Fingerprint generation time . . . 85
3.12 Traffic profile of the LatSeq showcasing experiment . . . 89
3.13 Waterfall of a burst of transmission in the uplink . . . 90

4.1 Ping RTT for different packet size . . . 96
4.2 Packet journey on the uplink for a 4.500-bytes packet . . . 99
4.3 Ping RTT in a loaded cell . . . 100
4.4 Time components of the HTTP completion time . . . 102
4.5 Webpage HTTP tranfer . . . 103
4.6 Flent Real-time RRUL experiment . . . 105
4.7 Flent RRUL downlink experiment analysis . . . 106
4.8 Flent RRUL uplink experiment analysis . . . 107
4.9 Uplink experiment transfer scenario . . . 108
4.10 Baseline uplink experiment results . . . 109
4.11 Commercial network experimental setup . . . 111
4.12 Evolution of the achieved throughput in the cell all along the day . . . 112
4.13 Uplink traffic profile in empty commercial network . . . 113
4.14 Uplink traffic profile in loaded commercial network . . . 114
4.15 Grant-based uplink access dynamic . . . 116
4.16 Short SR periodicity results . . . 118
4.17 Short BSR periodicity results . . . 120
4.18 Full bandwidth allocation to an explicit scheduling request results . . . 121
4.19 ECDF of latency per RAN configuration . . . 122
4.20 Buffer reported versus actual buffer status . . . 123
4.21 Uplink traffic profile observed by Park et al. . . . 125
4.22 Uplink traffic profile reported by Johansson . . . 126

5.1 TCP Cubic instantaneous throughput compared the available LTE network capacity . . . 129
5.2 ACK packet generation frequency . . . 130
5.3 Measured RTT by the TCP sender from ACKs over time . . . 131
5.4 Zoom on TCP traffic profile for a typical transmission cycle . . . 131
5.5 Experienced TCP RTT per packet . . . 133

5.6 Achieved throughput for a TCP Cubic transfer of 1 MB and 10 MB . . . . 134
5.7 RAN capacity utilization . . . . . 135

6.1 RTT and uplink traffic profile of a TCP downlink transmission in a commercial network . . . . . 139
6.2 Uplink MAC scheduler with Enhanced-BSR . . . . . 142
6.3 eBSR estimation model state machine . . . . . 143
6.4 ECDF RTT with and without eBSR . . . . . 148
6.5 eBSR results on uplink traffic profile . . . . . 149
6.6 Double feedback loop with TCP and eBSR . . . . . 151
6.7 TCP BBR downlink transfer results with eBSR . . . . . 151

7.1 Uplink dynamic scheduling model including eBSR prediction . . . . . 156
7.2 Classification of scheduling strategies . . . . . 157
7.3 Latency for different values of Fast-UL periodicity . . . . . 158
7.4 Adaptive periodicity of transmission opportunities . . . . . 160
7.5 MCS traces from a commercial network . . . . . 164
7.6 Uplink scheduling results for variable number of users and for different bandwidths . . . . . 165
7.7 Uplink delay evolution results . . . . . 166
7.8 Uplink OWD with distribution technique . . . . . 167
A.1 Random Access procedure . . . . . 176
A.2 RRC state machine. . . . . 178
A.3 UE energy consumption over time *(Source: ShareTechnote.com)* . . . . . 179
A.4 HTTP(S) request time-sequence . . . . . 180
B.1 DRB configurations . . . . . 181
C.1 LatSeq measurement part lifetime . . . . . 182
C.2 Assembly code of a LatSeq measurement point . . . . . 183
C.3 Assembly code of a LatSeq data collector . . . . . 184
C.4 LatSeq measurement point in the 3GPP stack . . . . . 185
C.5 Radio context representation . . . . . 186
C.6 CPU usage of LatSeq thread . . . . . 187
D.1 HTTPS transfer related traces . . . . . 188

E.1 Enhanced-BSR estimation dynamic.*Light gray line: estimated datarate; Gray area: estimated buffer size; Star: SR reception; Cross: received BSR value; Pulse line: UG; Dashed pulse line: received TBS.* . . . . . 194
E.2 Adaption of the model's parameters . . . . . 195
F.1 Simulation system model . . . . . 196
F.2 alloc_sim simulation testbed . . . . . 199

# LISTINGS


3.1 Consecutive LatSeq fingerprint of a packet segment . . . . . 67
3.2 measurement point code . . . . . 71
3.3 rdtsc synchronization fingerprint . . . . . 73
3.4 Assembly code to copy a data identifier . . . . . 74
3.5 Consecutive LatSeq fingerprint of a packet segment . . . . . 77
3.6 IP ID catching at the GTP layer . . . . . 78
3.7 Radio context capture with LatSeq "I" fingerprints . . . . . 78
3.8 Rebuilding output of a journey in a JSON format . . . . . 81
D.2 Define the activity of a UE . . . . . 190
D.2 Trigger a grants at the reception of a new scheduling request . . . . . 190
D.3 Update buffer estimation at the reception of a new BSR . . . . . 190
D.4 Update the buffer estimation at the reception of a new transport block . . 191
D.5 Filter UEs to be scheduled . . . . . 191

# List of Tables

1.1 SR and BSR comparison . . . . . 22
1.2 Techniques for reducing latency in 5G . . . . . 35

2.1 Solutions for cellular network research testbed . . . . . 46
2.2 OAI lab testbed configurations. . . . . 49

3.1 Implementation challenges for LatSeq measurement module . . . . . 71
3.2 LatSeq impact on BS performance results . . . . . 84

4.1 Summarized latency experiments. . . . . 95
4.2 User plane one-way delay in LTE RAN . . . . . 97
4.3 Commercial network testbed configurations. . . . . 111

6.1 Summarized enhanced-BSR evaluation results . . . . . 147

7.1 Summarized uplink scheduling results . . . . . 165

# List of Symbols

| Symbol | Unit | Meaning |
|---|---|---|
| $t$ | | Time |
| $i$ | | Index for user |
| $k$ | | Index for radio resource block |
| $\mu$ | | Numerology |
| $\mathcal{K}$ | | Set of radio resource blocks |
| $\mathcal{U}$ | | Set of user active the cell |
| $U$ | | Number of UEs active in the cell |
| $U_t^{act}$ | | Number of users active in the TTI $t$ |
| $\bar{U}^{act}$ | | Mean number of user active in a TTI |
| $Q_t^i$ | byte | Queue size at time $t$ for user $i$ |
| $Q^{i,\mathrm{bsr}}$ | bytes | Queue size reported by BSR by user $i$ |
| $\hat{Q}_t^i$ | byte | Estimated queue size at time $t$ |
| $T_{\mathrm{tti}}$ | ms | Duration of a TTI = 1 ms (LTE) |
| $T_{\mathrm{harq}}$ | ms | HARQ feedback loop delay = 8 ms (LTE) |
| $T_{\mathrm{bsr}}$ | ms | Expected interval between 2 BSRs e.g. = 64 ms |
| $T_{\mathrm{sr}}$ | ms | Expected interval between 2 SRs e.g. = 10 ms |
| $T_{\mathrm{tx}}$ | ms | Transmission and processing time of UE e.g. = 4 ms (LTE) |
| $T_{\mathrm{sched}}$ | ms | Scheduling process time e.g. = 4 ms (LTE) |
| $T_Q$ | ms | Queuing delay |
| $T^{i,act}$ | ms | Number of consecutive TTIs during which UE $i$ could transmit |
| $T^{i,cycle}$ | ms | Length of transmission cycle for the UE $i$ |
| $\lambda_{i,k}$ | | Generic metric representing scheduling algorithm for user $i$ on radio resource $k$ |
| $d_t^i$ | bps | Datarate achieved for user $i$ at time $t$ |
| $d_t^{i,k}$ | bps | Datarate achievable for user $i$ on RB $k$ |
| $\bar{d}_t^i$ | bps | Mean datarate achieved by user $i$ before $t$ |
| $d_t^{i,B}$ | bps | Datarate achievable by user $i$ if it uses all the bandwidth $B$ |

| | | |
|---|---|---|
| $D^i$ | bps | Datarate generated at time $t$ by user |
| $\hat{D}_t^i$ | bps | Estimated uplink datarate at time $t$ |
| $g$ | bytes | Grant size |
| $L$ | bytes | Number of bytes transmitted |
| $B$ | MHz | Bandwidth |
| $\rho_n$ | % | Transport block utilization |
| $\xi_{\text{cell}}$ | % | Cell load |
| $P_{con}$ | W | Power consumption when the UE is connected and in active state |
| $P_{idle}$ | W | Power consumption associated to idle state when UE is connected but not active |
| $P_{Rx+Tx}$ | W | Power to send or receive a signal |
| $P_{Tx}$ | W | Power for transmission module |
| $P_{TxRF}$ | W | Power consumption associated to RF transmission |
| $P_{TxBB}$ | W | Power consumption to encode at a bitrate |

# List of accronyms

| | |
|---|---|
| 3GPP | Third Generation Partnership Project |
| 5GC | 5G Core |
| 5QI | 5 QoS Identifier |
| AAS | Advanced Antenna System |
| ACK | Acknowledgment |
| ADB | Android Data Bridge |
| AI | Artificial Intelligence |
| AM | Acknowledged Mode |
| AMC | Adaptive Modulation and Coding |
| AMF | Access and Mobility Management Function |
| API | Application Programming Interface |
| AQM | Active Queue Management |
| ARED | Adaptive Random Early Detection |
| ARQ | Automatic Repeat reQuest |
| BBR | Bottleneck Bandwidth and RTT |
| BCQI | Best-CQI / Max-Rate |
| BDP | Bandwidth-Delay Product |
| BE | Best Effort |
| BIF | Bytes-in-Flight |
| BLER | Block Error Rate |
| BS | Base Station |
| BSR | Buffer Status Report |
| C-RAN | Cloud-RAN |

CB . . . . . Code-Block

CBG . . . . . Code-Block Group

CBR . . . . . Constant BitRate

CCA . . . . . Congestion Control Algorithm

CDF . . . . . Cumulative Distribution Function

CDN . . . . . Content Delivery Network

CoDel . . . . . Controlled Delay

COTS . . . . . Commercial Off-The-Shelf

CP . . . . . Control-Plane

CPU . . . . . Central Processing Unit

CQI . . . . . Channel Quality Indication

CU . . . . . Centralized Unit

CWND . . . . . Congestion Window

DCE . . . . . Direct Code Execution

DCI . . . . . Downlink Control Information

DRB . . . . . Data Radio Bearer

DRX . . . . . Discontinuous Reception

DU . . . . . Distributed Unit

E-UTRA . . . . . Evolved Universal Terrestrial Radio Access

ECDF . . . . . Empirical Cumulative Distribution Function

ECN . . . . . Explicit Congestion Notification

eMBB . . . . . enhanced Mobile BroadBand

eNB . . . . . Evolved NodeB

EPC . . . . . Evolved Packet Core

EXP/PF . . . . . Exponential/PF

Fast-UL . . . . . Fast Uplink Grant

FD . . . . . Frequency-Domain

FDD . . . . . Frequency-Division Duplexing

FEC . . . . . Forward Error Correction

FIFO . . . . . First-In First-Out

FPGA . . . . . Field Programmable Gate Arrays

gNB . . . . . gNodeB

GTP . . . . . GPRS Tunneling Protocol

HARQ . . . . . Hybrid-ARQ

HoL . . . . . Head-of-Line

HSS . . . . . Home Subscriber Server

HTTP . . . . . HyperText Transfer Protocol

ICMP . . . . . Internet Control Message Protocol

IETF . . . . . Internet Engineering Task Force

IP . . . . . Internet Protocol

IPPM . . . . . IP Performance Measurement group

IR . . . . . Incremental Redundancy

ITU . . . . . International Telecommunication Union

JSON . . . . . JavaScript Object Notation

KPI . . . . . Key Performance Indicator

L4S . . . . . Low Latency, Low Loss, Scalable throughput

LA . . . . . Link Adaptation

LCID . . . . . Logical Channel IDentifier

LCP . . . . . Logical Channel prioritization

LDPC . . . . . Low-Density Parity-Check Code

LTE . . . . . Long Term Evolution

MAC . . . . . Medium Access Control

MAC CE . . . . . MAC Control Element

MCS . . . . . Modulation and Coding Scheme
MEC . . . . . Mobile Edge Computing
MIMO . . . . . Multiple-Input Multiple-Output
MME . . . . . Mobile Management Entity
mMTC . . . . . massive Machine Type Communications
MNC . . . . . Mobile Network Core
MTU . . . . . Maximum Transmission Unit

NG-RAN . . . . . Next Generation RAN
NIC . . . . . Network Interface Card
NR . . . . . New Radio
NSA . . . . . Non-StandAlone mode
NTP . . . . . Network Time Protocol

OAI . . . . . OpenAirInterface
OFDM . . . . . Orthogonal Frequency-Division Multiple
OFDMA . . . . . Orthogonal Frequency-Division Multiple Access
OS . . . . . Operating System
OSA . . . . . OpenAirInterface Software Alliance
OSI . . . . . Open Systems Interconnection
OWD . . . . . One-Way Delay

PC . . . . . Power Control
PDCCH . . . . . Physical Downlink Control Channel
PDCP . . . . . Packet Data Convergence Protocol
PDN . . . . . Packet Data Network
PDU . . . . . Protocol Data Unit
PDV . . . . . Packet Delay Variation
PF . . . . . Proportional-Fair
PGW . . . . . Packet Gateway

PHR . . . . . Power Headroom

PHY . . . . . Physical Layer

PIE . . . . . Proportional Integral controller Enhanced

PRB . . . . . Physical Resource Block

PUCCH . . . . . Physical Uplink Control Channel

QAM . . . . . Quadrature-Amplitude

QCI . . . . . QoS Class Identifier

QoS . . . . . Quality of Service

QPSK . . . . . Quadrature Phase Shift Keying

QUIC . . . . . Quic

RACH . . . . . Random Access procedure Channel

RAN . . . . . Radio Access Network

RAT . . . . . Radio Access Technology

RB . . . . . Resource Block

RF . . . . . Radio-Frequency

RFC . . . . . Request For Comments

RLC . . . . . Radio Link Control

RNTI . . . . . Radio Network Temporary Identifier

ROHC . . . . . Robust Header Compression

RR . . . . . Round-Robin

RRC . . . . . Radio Resource Control

RRH . . . . . Remote Radio Head

RRUL . . . . . Realtime Response Under Load

RTO . . . . . Retransmission TimeOut

RTT . . . . . Round-Time Trip

SA . . . . . Standalone

SC-FDMA . . . . . Single-Carrier FDMA

SD-RAN . . . . . Software-Defined RAN
SDAP . . . . . Service Data Adaptation Protocol
SDR . . . . . Software-Defined Radio
SDU . . . . . Service Data Unit
SGW . . . . . Serving Gateway
SISO . . . . . Single-Input Single-Output
SLOC . . . . . Source Lines of Code
SN . . . . . Sequence Number
SNR . . . . . Signal-to-Noise Ratio
SPS . . . . . Semi-Persistent Scheduling
SR . . . . . Scheduling Request
SRB . . . . . Signal Radio Bearer
SRS . . . . . Sounding Reference Signal

TB . . . . . Transport Block
TBS . . . . . Transport Block Size
TCP . . . . . Transmission Control Protocol
TD . . . . . Time-Domain
TDD . . . . . Time-Division Duplexing
TLS . . . . . Transport Layer Security
TM . . . . . Transparent Mode
TTI . . . . . Transmission Time Interval
TxOp . . . . . Transmission Opportunity

UDP . . . . . User Datagram Protocol
UE . . . . . User Equipment
UG . . . . . Uplink Grant or grant of uplink capacity
UL-SCH . . . . . Uplink Shared Channel
UM . . . . . Unacknowledged Mode
UPF . . . . . User Plane Function

URLLC . . . . . Ultra-Reliable Low-Latency Communications

USRP . . . . . Universal Software Radio Peripheral

VoLTE . . . . . Voice over LTE

VR . . . . . Virtual Reality

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

**Titre :** Réduction de la Latence et de la Gigue dans le Réseau d'Accès Radio 5G

**Mot clés :** 5G, Latence, Gigue, Mesure, OpenAirInterface, Ordonnancement remontant

**Résumé :** Les réseaux cellulaires présentent des latences importantes, en particulier au niveau du réseaux d'accès radio (RAN). La nouvelle génération de technologie cellulaire 5G, en plus d'augmenter les débits de transmission, doit offrir de faibles latences en connectivité mobile.
L'objectif de cette thèse est de proposer des mécanismes de réduction de la latence dans le RAN. Dans cette perspective, nous avons mis en place une plateforme d'expérimentation software-defined RAN basée sur le projet OpenAirInterface et conçu un outil de mesure fine des latences internes au RAN appelé LatSeq. Nous montrons que la latence aller-retour atteint typiquement 30 ms en LTE répartie entre délais de traitement, de mis en file d'attente, de retransmissions et d'acquisition du medium. Nous avons aussi mesuré une gigue importante sur la voie remontante qui influe sur la latence du RAN mais aussi sur les performances des trafics TCP, très répandus sur Internet.

LatSeq met en évidence le rôle du mécanisme d'allocation des ressources radio dans la gigue et la latence de la voie remontante. Nous montrons comment la connaissance imparfaite de la taille du buffer de transmission par l'algorithme d'ordonnancement à partir des rapports Buffer Status Reports (BSR) et des demandes d'accès (SR) induit une transmission par rafale, une gigue importante et une sous-utilisation de la capacité du canal radio.

Nous proposons alors un nouveau mécanisme d'estimation de la taille du buffer de transmission des terminaux compatible avec les standards 3GPP LTE et 5G, et démontrons que cette méthode permet de réduire la latence de la voie remontante et la gigue subie par les flux de paquets.

**Title:** Reducing Latency and Jitter in the 5G Radio Access Network

**Keywords:** 5G, Latency, Jitter, Measurements, OpenAirInterface, Uplink scheduler

**Abstract:** Cellular networks present high latencies, especially in the Radio Acccess Network (RAN). The new generation of cellular technology 5G, in addition to increasing transmission throughput, must offer low latency in mobile connectivity.
The objective of this thesis is to propose mechanisms for reducing latency in the RAN. In this perspective, we set up a software-defined RAN experimentation platform based on the OpenAirInterface project and designed a tool for fine-grained measurement of RAN internal latencies called LatSeq. We show that the round-time trip latency typically reaches 30 ms in LTE divided between processing, queueing, retransmission and medium acquisition delays. We also measure a significant jitter on the uplink that affects the RAN latency but also the performance of TCP traffics, which are very common on the Internet.

LatSeq highlights the role of the radio resource allocation mechanism in the jitter and latency of the uplink channel. We show how the partial knowledge of the transmission buffer length by the scheduling algorithm from Buffer Status Reports (BSRs) and Scheduling Requests (SRs) induces bursts of transmissions, significant jitter and under-utilization of the radio channel capacity.

We then propose a new mechanism for estimating the terminal transmission buffer length which is compatible with 3GPP LTE and 5G standards. We demonstrate that this method makes it possible to reduce the latency of the uplink channel and even more the jitter experienced by packet flows.