SMARTPHONES ARE THE KEY!

On the constraints of mobile computing in cloud avoidance

RÉMY RAES





MADIS doctoral school
Inria centre at the University of Lille
CRIStAL — Spirals & WIDE research teams

Thesis submitted and defended on July 6th, 2026
for the grade of Doctor in Computer Science

Supervisors

Romain Rouvoy	Professeur des universités	University of Lille
François Taïani	Professeur des universités	University of Rennes
Davide Frey	Chargé de recherche	Inria Rennes
Adrien Luxey-Bitri	Maître de conférences	University of Lille

Reviewers

Hélène Coullon	Maîtresse de conférences (HDR)	IMT Atlantique
Antoine Boutet	Maître de conférences (HDR)	INSA Lyon

Examiners

Sonia Ben Mokhtar	Directrice de recherche	CNRS Lyon (jury president)
Stéphane Delbruel	Maître de conférences	University of Bordeaux

LES SMARTPHONES SONT LA CLÉ !

Des contraintes de l'informatique mobile dans l'évitement du cloud

RÉMY RAES



Ecole doctorale **MADIS**
Centre Inria de l'Université de Lille
CRIStAL — Equipes-projet Spirals & WIDE

Thèse présentée et soutenue le 06/07/2026
pour obtenir le grade de docteur en informatique

Directeurs et encadrants

Romain Rouvoy	Professeur des universités	Université de Lille
François Taïani	Professeur des universités	Université de Rennes
Davide Frey	Chargé de recherche	Inria Rennes
Adrien Luxey-Bitri	Maître de conférences	Université de Lille

Rapporteurs

Hélène Coullon	Maîtresse de conférences (HDR)	IMT Atlantique
Antoine Boutet	Maître de conférences (HDR)	INSA Lyon

Examinateurs

Sonia Ben Mokhtar	Directrice de recherche	CNRS Lyon (présidente du jury)
Stéphane Delbruel	Maître de conférences	Université de Bordeaux

Rémy Raes: *Smartphones are the key!*, On the constraints of mobile computing in cloud avoidance, © July 2026

ABSTRACT

Since the introduction of the first iPhone in 2007, smartphones changed the way we access and generate information, leading to major economic and societal changes, including for instance the rise of "Big Tech" companies and the shift of traditional media to social networks.

In this context, my thesis explores the general question of control and autonomy left to users, notably in terms of privacy (and its implications on both data storage and communication) and energy efficiency. I propose to explore three key, complementary questions linked to this broad issue:

1. **Local storage and protection of data.** Our smartphones produce data that is both valuable and private, such as location data. We propose an integrated suite of tools to enable local storage of big data and privacy checks, allowing local verification of personal data before sharing it with peers.

2. **Bypassing the centralized intermediate actors through direct terminal-to-terminal communication.** Despite the originally decentralized Internet model, communications now widely revolve around data centres. We explore how this communication model can be challenged, presenting the results of a survey, then propose a tool to help developers adopt local communication.

3. **The impact of hardware and software choices on the energy consumption of smartphones.** Heavy data processing impacts the battery of smartphones, and thus the end user experience. We propose an evaluation protocol allowing Android developers to improve their apps through library selection, and subsequently propose a power model able to determine the repartition of the power draw between smartphone components.

RÉSUMÉ

Depuis l'arrivée du premier iPhone en 2007, les smartphones ont fondamentalement changé la façon dont nous consommons et créons de l'information, amenant des changements économiques et sociétaux majeurs, incluant par exemple la montée de "géants de la tech" et le passage des médias traditionnels aux réseaux sociaux.

Dans ce contexte, ma thèse s'intéresse à la question générale du contrôle et de l'autonomie laissés aux utilisateurs, notamment en termes de vie privée (et de ses implications sur le stockage et la communication des données) et d'efficacité énergétique. J'y propose d'étudier trois questions clés liées à cette problématique générale :

1. **La protection et le stockage locaux des données privées.** Nos smartphones produisent des données qui sont à la fois précieuses et privées, comme les données de géolocalisation par exemple. Nous proposons une suite d'outils permettant le stockage de grandes masses de données et le contrôle de leur confidentialité, tous deux sur smartphone, permettant ainsi la vérification locale des données à caractère personnel avant leur partage avec des tiers.

2. **Le contournement des intermédiaires centralisés par des communications terminal-à-terminal.** Malgré le modèle initialement décentralisé d'Internet, les communications sont majoritairement dépendantes des centres de données. Nous examinons comment ce modèle de communication peut être remis en question, en présentant les résultats d'une enquête menée auprès de développeurs, puis en proposant un outil destiné à aider ces derniers à adopter une communication plus locale entre smartphones.

3. **L'impact des choix logiciels et matériels sur la consommation énergétiques des smartphones.** Les traitements de données sur smartphone impactent leur batterie, et donc l'expérience utilisateur. Nous proposons un protocole d'évaluation permettant aux développeurs Android d'améliorer leurs applications via la sélection de librairies plus efficientes, puis proposons un modèle capable d'identifier la répartition de la consommation énergétique d'un smartphone entre ses différents composants.

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

PEER-REVIEWED INTERNATIONAL JOURNALS

[3] Emilio Molina, Mirko Fiacchini, Arthur Goarant, Rémy Raes, Sophie Cerf, and Bogdan Robu. "Application of a predictive method to protect privacy of mobility data." In: *Control Engineering Practice* 156 (Mar. 2025), p. 106223. DOI: 10.1016/j.conengprac.2024.106223. URL: https://hal.science/hal-04885266.

[6] Rémy Raes, Olivier Ruas, Adrien Luxey-Bitri, and Romain Rouvoy. "INTACT: Compact Storage of Data Streams in Mobile Devices to Unlock User Privacy at the Edge." In: *Journal of Internet Services and Applications (JISA)* 16.1 (June 2025), 372–387. DOI: 10.5753/jisa.2025.5242. URL: https://journals-sol.sbc.org.br/index.php/jisa/article/view/5242.

[7] Rubén Saborido, Rémy Raes, Rodrigo Morales, Romain Rouvoy, Foutse Khomh, and Yann-Gaël Guéhéneuc. "Simply the Best -A Systematic Evaluation Approach for Third-Party Libraries based on Mobile App Quality Attributes." In: *Empirical Software Engineering* 31.5 (May 2026), p. 137. DOI: 10.1007/s10664-026-10872-w. URL: https://link.springer.com/article/10.1007/s10664-026-10872-w.

PEER-REVIEWED INTERNATIONAL CONFERENCES

[1] Lauric Desauw, Adrien Luxey-Bitri, Rémy Raes, Romain Rouvoy, Olivier Ruas, and Walter Rudametkin. "A Critical Review of Mobile Device-to-Device Communication." In: *Proceedings of the 25th International Conference on Distributed Applications and Interoperable Systems (DAIS'25)*. Ed. by Daniel Balouek and Ibéria Medeiros. LNCS. Springer, June 2025, pp. 1–24. ISBN: 978-3-031-95728-4. DOI: 10.1007/978-3-031-95728-4_1. URL: https://inria.hal.science/hal-04198528v1.

[2] Édouard Guégain, Rémy Raes, Noé Chachignot, Clément Quinton, and Romain Rouvoy. "AndroWatts: Unpacking the Power Consumption of Mobile Device's Components." In: *Proceedings of the 12th International Conference on Mobile Software Engineering and Systems (MOBILESoft'25)*. Ottawa, Canada, Apr.

2025. DOI: 10.1109/MOBILESoft66462.2025.00015. URL: https://hal.science/hal-04928609.

[4] Rémy Raes, Adrien Luxey-Bitri, Romain Rouvoy, Davide Frey, and François Taïani. "Venice: eschewing the cloud by leveraging local communication channels." In: *Joint Proceedings of ICT4S 2024 Doctoral Symposium, Demonstrations & Posters Track and Workshops*. Stockholm, Sweden, June 2024, pp. 1–4. URL: https://hal.science/hal-04576743.

[5] Rémy Raes, Olivier Ruas, Adrien Luxey-Bitri, and Romain Rouvoy. "Compact Storage of Data Streams in Mobile Devices." In: *Proceedings of the 24th International Conference on Distributed Applications and Interoperable Systems (DAIS'24)*. LNCS. Groningen, Netherlands: Springer, June 2024. DOI: 10.1007/978-3-031-62638-8_4. URL: https://hal.science/hal-04535716.

ACKNOWLEDGMENTS

Disclaimer: Because I know much of the people mentioned here does not speak English, I will switch to French for this section.

(Avant de rentrer dans le dur, s'il vous plaît, ne vous sentez pas lésé·e si votre nom n'apparaît pas dans cette section : il est impossible de lister exhaustivement les personnes qui ont eu un impact positif sur ma vie pendant ces trois années de thèse.
Si c'était toutefois le cas, n'hésitez pas à m'envoyer un mail (contact [AT] remyraes [DOT] com), j'envisagerai alors la possibilité de vous envoyer une carte postale de welsh pour me faire pardonner.)

AUX MEMBRES DE MON JURY. **Hélène**, merci d'avoir accepté de rapporter ma thèse, d'autant plus que j'avais l'impression de te faire un cadeau empoisonné vu la largeur de mes recherches ; c'est avec plaisir que nous reparlerons d'edge computing à l'occasion. **Antoine**, merci également pour ton rapport de manuscrit ; j'ai hâte de rejoindre ton équipe pour poursuivre mes recherches mêlant mobile et vie privée. **Stéphane**, merci d'avoir accepté d'examiner ma défense de thèse ; on m'a dit que j'avais été plutôt abrupt dans certaines de mes réponses à tes questions, j'avoue avoir un peu anticipé ta franchise, nous aurons je l'espère encore l'occasion d'en discuter, que ça soit dans un bureau, autour d'une bière ou dans un ascenseur ! Enfin, **Sonia**, merci d'avoir présidé mon jury de thèse (même si tu semblais ne pas être au courant de cette responsabilité au début de la soutenance) ; j'ai particulièrement apprécié notre discussion sur l'apprentissage décentralisé, et la poursuivrai avec plaisir si d'aventure nous nous croisions à Lyon ou ailleurs.

Une seconde salve de remerciements toute particulière à **Antoine** et **Stéphane**, qui m'avez accueilli dans l'équipe d'organisation de trois écoles d'hiver : merci pour nos discussions jusqu'à parfois tard le soir, qui m'ont permis de mieux comprendre le côté humain, collaboratif de ce qu'est la recherche.

À MES ENCADRANTS. **Romain**, mille mercis ; un jour peut-être, je saurai être aussi multi-tâches (à donner des cours à la fac, diriger le master informatique et les recherches de 10 doctorants en parallèle) et pourtant systématiquement pertinent dans mes retours à tous mes collègues, que tu ne l'as été pendant ces trois années de thèse. Merci **François** pour avoir cru en moi dans cette aventure, merci pour ta gentillesse, ta bienveillance et ta patience, toutes salvatrices dans les temps difficiles de la recherche. **Davide**, merci pour ta franchise (je me

souviens notamment de notre rush sur le papier ICDCS pendant et après les vacances de Noël, où tu m'as bien poussé à bout !) : la remise en question permanente contribue effectivement à la construction de travaux solides. Enfin, merci **Adrien** pour le partage du bureau, pour ton engagement associatif, pour ton point de vue parfois décalé, souvent militant sur le monde et la place de la recherche dans la société.

À SPIRALS, MON ÉQUIPE. **Lionel**, merci pour ta direction sans faille de l'équipe qui lui permet de faire son boulot dans les meilleures conditions et ambiance possibles. **Laurence**, je te croise étrangement moins depuis que tu es devenue doyenne de la faculté d'informatique ; je n'oublie cependant pas ton énergie, ni les temps troublés du premier confinement en 2020 où tu m'as invité un midi chez vous partager un moment, un repas, et un délicieux limoncello maison.

Pierre (J.), ce fut un plaisir de partager des frites à Lille, des bières à Paris et des moments de vie au Pleynet. Rendez-vous à Lyon ? Merci **Walter** pour ta non-retenue et ta bonne humeur à toute épreuve. Merci à mon "vieux" SPIRALS : **Salman, Antonin, Arthur, Jordan, les 2 Antoine, Olivier, Jean-Luc, Maryam, Giuseppe, Edouard, Sarra, Alexandre, Marion**, vous avez contribué à me donner l'envie de faire cette thèse durant mes années d'ingénieur dans l'équipe.

Plus proche du SPIRALS de 2026, merci **Iliana** pour les karaokés, les dons du sang, l'ADSL, ta curiosité et ton empathie sans limites, tu es un exemple à bien des égards pour moi. **Daniel**, merci d'avoir accepté de m'accompagner dans l'enfer technique du device-to-device, et plus important encore, de gérer le déplacement de l'équipe vers la cafétéria tous les midis. Merci **Pierre (L.)** et **Clémentine**, c'est toujours un plaisir de partager un moment avec vous, que ça soit autour d'un café au labo, d'une flûte de champagne à l'Hospice Gantois, d'une gaufre à Dunkerque, d'un billet de ciné à l'UGC ou autour d'un sac de pellets. **Simon**, merci pour nos discussions, et merci de m'avoir inclus dans l'organisation de DisCoTec 2025 à Lille, ce fut un plaisir de pouvoir te filer un coup de main. Merci **Pierre (B.), Sophie, Olga, Clément** pour votre présence rassurante et vos conseils précieux sur le déroulement et la poursuite d'une carrière scientifique. Dans un désordre complet, merci à **Thibaud, Noé, Gaëlle, Belkis, Nada, Sihem, Kellian, François, Jérémy, Samuel** pour la bonne ambiance au bureau.

Merci à mes voisins de bureau de RMOD (désormais EVREF) : **Larisa**, **Imen**, **Marius**, je n'ai jamais été très bon en small talk, mais nos petites discussions au détour d'un couloir m'ont toujours remonté le moral. Merci aux assistantes de l'équipe, **Karine** et **Isabelle**, pour votre soutien de valeur dans l'établissement des missions ainsi que dans le montage financier des événements. Enfin, merci à **Giovanni** et **Aurélien** pour votre direction du défi Inria FedMalin, lequel a financé ma thèse.

À WIDE, MON (AUTRE) ÉQUIPE. Merci à **Jade, Manon, Timothée, Mathieu, Hugo, Djob, Caleb, Adrien, Opale, Dimitri, Amélie, Césaire, Cyrille,** pour votre accueil toujours chaleureux lors de mes (trop) rares visites à Rennes, entre cafés, bières, films douteux, pizzas, séminaires à la mer et galettes bretonnes.

À MES CAMARADES DE LA MUSIQUE. La musique s'avérant occuper une grande partie de ma vie, il est possible que ce paragraphe soit le plus long de cette section. Déso pas déso.

Merci à mes amies et amis de la Musique Municipale d'Erquinghem-Lys, avec lesquels j'ai passé plus de 15 ans à voyager entre différents pupitres, vous m'avez appris et apporté beaucoup. **Jean-Noël**, c'est à ta patience tout au long de ma formation (depuis mes 6 ans !) et à ta bonne humeur que je dois mon amour de la musique. Merci **Pascal** de m'avoir accueilli au pupitre des tubas, et m'avoir donné envie de faire du souba un peu plus sérieusement. **Bernard** et **Yves**, merci pour votre direction (pas toujours facile) de l'harmonie, pour votre investissement dans les répétitions et la création des concerts, et pour m'avoir poussé à prendre la baguette de temps en temps. Merci **Cyril** et **Virgile** pour les multiples restos et discussions. Enfin, merci à toutes les copines et les copains des Chouff'eud'in, **Sébastien, Stéphane, Jules, Lucas, Simon, Camille, Romain, Lucile, Zoé, Louise** : vous avez été mon premier groupe de rue, et c'est grâce à vous que j'y ai pris goût ! Merci pour nos pérégrinations musicales qui, de braderie en course à pied, nous auront emmené jusqu'à la grande scène du festival de bandas de Condom.

Merci à la Fanfare Communale de Rexpoëde, votre soutien à travers les confinements et la maladie m'ont été salvateurs. Je pense notamment à **Sam** pour tes conseils au tambour, et à **Hervé** et **Alex** pour votre bonne humeur constante ; un grand merci, et encore désolé d'avoir dû partir.

Cécile, Gonzague, Joseph, Loudj, Axel, Tuang et **Marine** de Fanfarena21, je pensais pas que rock et fanfare étaient compatibles, merci de m'avoir montré le contraire et de m'avoir accueilli parmi vous pour partager du bon temps (c'était une super idée de faire le festival des Festifuries la veille de ma défense de thèse).

Simon, merci d'être force de propositions musicales pour tous les groupes auxquels je contribue, merci pour les bières, les frites à l'Annexe et pour ton oreille qui y était parfois attentive (on va le faire cet album des méchants !).

Attaquons maintenant le plus gros morceau : la Frontale.

Titi, merci pour ton assiduité à la percussion, ton énergie et la bonne humeur que tu instilles à chaque répét et dans chaque contrat (tu le feras ce set à la caisse claire, promis !) (mais appelle-moi avant qu'on bloque une date stp) (peut-être que je répondrai, cette fois). **Estelle**, merci pour les questions existentielles. **Marion**, merci pour ton sourire

débridé et nos vacances partagées : à l'ascension du Vercors ou bien tout au fond du Cantal, c'est à chaque fois la déconnexion assurée. **Louis**, merci pour ta direction, ton travail sans relâche, ta dévotion, ton courage et pour les bières. **Jérôme** et **Aurore**, merci pour vos mots justes, votre présence et votre soutien dans les moments difficiles. **Toune**, merci pour ton grand cœur et ton humanité. **David**, jamais dans ma vie n'aurais-je pensé apprécier un jacuzzi de minuit à boire du limoncello et fumer des cigares, le tout pas très habillé ; merci pour le souba et les stouts. **Juliette, Alex** et **Léa**, merci pour les soirées SNK et un max de bières en toute décontraction. **Matthieu**, c'est et ça sera toujours un plaisir de discuter avec toi autour d'Orvaux, et de chanter du Jean-Jacques au karaoké. **Sasha**, c'est pas bien de nous faire nous coucher à 6h du matin ; merci tout de même pour les bières, le Shanghai et les chauve-souris.

2025 a été une année particulière pour la Frontale, car nous y avons accueilli une sacrée bande de loustics, toutes prêtes et tous prêts à "prendre la relève" des "vieux" de la fanfare.

Sixtine, fais-toi confiance, merde ! Merci pour ton sourire, ton travail au trombone, ta sincérité et ta curiosité sans limites qui (semblent te faire apparaître aux yeux des autres comme très frontale voire folle, mais) me sont infiniment rafraîchissants. **Laure, Noémie, Camille, Constance, Philippine, Emilie, Ava,** ce fut un plaisir immense de chanter avec vous (tant que c'est Ghost et pas les démons de minuit).

Merci **Milène** de m'avoir introduit à cette formation musicale si particulière, de l'avoir présidée pendant un an et de m'avoir supporté dans ma direction ; merci également à **Pipo** d'être l'exact opposé de Milène et de mettre un bordel monstre à chaque prestation. Mention spéciale pour **Maxime**, mentor et camarade de percussion : merci pour ton sourire à toute épreuve, c'est toujours un grand plaisir de jouer en duo avec toi et de perdre un maximum de gens en cours de morceau avec des patterns de cloche à la con.

Dans un désordre complet, merci **Arthur, Sarah, Alexis, Juliette, Pierre, Alice, Gaspard, Emma, Louise, Mathilde, Loïc, Hélène, Hélène, Corentin, Aude, Théophile, Antoine, Antoine, Garance, Eloïse, Claire, Paul, Zoé, Tom, Romane, Tanguy, Valentin, Bidon, Clara, Lucas, Manon, Pascal,** de faire la Frontale.

Enfin, **Nathan**, je t'aurais bien mentionné dans les remerciements à Spirals, mais je crois qu'on a passé plus de temps ensemble en fanfare qu'au labo ! Je me sens parfois (souvent !) responsable de la charge que tu as subi cette année en tant que président (après tout, c'est moi qui t'y ai invité), mais j'ai quand même l'impression que la Frontale a évolué dans le bon sens sous ta direction. Merci pour le sponsoring de Roncq Automobiles, pour l'organisation de vacances à la mer et à la montagne pour 36 musiciens, pour ta gestion parfois (souvent ?) pas facile du groupe, et pour tes patience et respect infinis envers tous.

À MES (AUTRES) CAMARADES. Merci **Son** pour un nombre incalculable de crêpes dévorées au vélo de Suzon après une Lego League, et pour supporter mon caractère de merde quand on joue ensemble à Overwatch. Merci **Pierre** pour ton grand cœur et ton soutien inconditionnel à travers les épreuves, alors que tu n'es toi-même pas épargné par la vie. Merci à vous deux, et merci également à **Algäé, Alix, Amalia, Armand, Charlotte, Fabien, Paul, Plume** et **Steven** pour le partage de semaines de vacances en Bourgogne, en Vendée, en Bretagne, dans les Landes, dans le Périgord, à Avoriaz... Ces parenthèses sont toujours très riches en événements, parfois dangereuses (même si à vrai dire, le baptême d'hélico était marrant), mais permettent toujours de déconnecter et me sont salvatrices. Merci **Nina** pour les belles pages de vie rédigées ensemble.

À MES COLOCS. **Alexia**, même si t'es pas officiellement ma coloc, merci pour tes sessions de saxophone à l'appart qui apaisent l'ambiance, pour nos discussions parfois animées et ta motivation à toujours essayer de nous entraîner Théo et moi dans des jeux de société que tu t'obstines décidément bien trop à perdre à chaque partie.

Doli, honnêtement, j'avais initialement peur de me mettre en colocation par crainte de perdre mon espace personnel, mais il s'avère que nos deux agendas de ministres musiciens m'ont permis de respecter ce besoin de solitude, tout en communiant socialement si le besoin s'en faisait ressentir. Merci pour toutes nos pérégrinations musicales communes, et pour ta cuisine, toujours réconfortante, et plus encore pendant la rédaction de ce manuscrit.

À MA FAMILLE. Merci à vous, qui je pense ne comprenez rien à mon boulot mais me supportez quand même.
Christine et Hubert pour tenir la famille à travers les épreuves. **Jean-Lou** pour absolument tout. **Julien, Héloïse, Chantal, Dominique, Mathis, Luis** et **Rafaël** pour les vacances en Vendée. **Alicia** et **Evan** pour votre accueil toujours chaleureux quand je suis de passage à Grenoble. **Chloé** qui me montre constamment qu'avec de la motivation on peut franchir des montagnes, et même rallier Colmar depuis Paris à pied ; **René-Jean** et **Maxime** pour me montrer qu'il est possible, et même presque pas si terrible que ça, de conduire un camping-car en première pendant tout le trajet. **Théo** et **Martin** pour m'avoir supporté pendant pas mal de temps, et pour nos sessions fraternelles Burger King quasi hebdomadaires ; on ne parle pas beaucoup, mais on se soutient.

À MON ÉTOILE.

Maman, ça m'emmerde profondément que tu ne sois plus là pour voir l'accomplissement de ce qu'on pourrait qualifier de jolie petite étape dans ma vie de chercheur. J'aime cependant penser que tu aurais été immensément fière de ton grand garçon.

Je pense à toi, toujours et pour toujours.
Bisous maman !

CONTENTS

ABSTRACT v
LIST OF PUBLICATIONS vii
LIST OF FIGURES xvi
LIST OF TABLES xx
LIST OF ACRONYMS xxii
1 INTRODUCTION 1
1.1 Motivation . . . 1
1.2 Contributions . . . 4
1.3 List of Scientific Publications . . . 5
1.4 Proofs of concept . . . 7
1.5 Outline . . . 7
2 BACKGROUND 9
2.1 Protecting user privacy through local data storage . . . 10
2.2 Unlocking local device-to-device communication . . . 13
2.3 Understanding and improving mobile energy usage . . . 17
2.4 Summary . . . 21
3 ONLINE DATA STORAGE THROUGH PIECEWISE LINEAR COMPRESSION 23
3.1 Compact storage of data streams . . . 24
3.2 A lossy storage middleware for multivariate time series 39
3.3 Protecting user privacy through local data storage . . . 51
3.4 Application of a predictive method to protect the privacy of mobility data . . . 58
3.5 Discussion . . . 59
4 LOCAL DEVICE-TO-DEVICE COMMUNICATION 65
4.1 An attractive, yet poorly adopted communication means 66
4.2 A communication framework to remove usage friction 83
4.3 Discussion . . . 86
5 ENERGY USAGE 89
5.1 Measuring and comparing the impact of TPLs on app quality . . . 90
5.2 Understand mobile hardware consumption repartition 102
5.3 Discussion . . . 106
6 CONCLUSION & PERSPECTIVES 111
6.1 Contribution . . . 111
6.2 Perspectives . . . 112

BIBLIOGRAPHY 115

I APPENDIX
A SERIES CLUSTERING FOR FLIND 141

Figure 2.1 For an inherently D2D use case such as transferring a file between nearby users, resorting to an Internet service for the exchange leads to more energy consumption, more storage space used, and exposes the users to digital surveillance (degraded privacy). 15

Figure 3.1 A time series $\mathcal{P}$ being modelled with Fast Linear Interpolation (FLI), displaying every component of FLI's data structure $\mathcal{F} = (\mathcal{H}, g_M, g_{\min}, g_{\max}, p_{\text{last}})$. $\mathcal{H}$ models every point in $\mathcal{P}$ while keeping their error below ϵ. $\mathcal{H}$'s last entry, $\mathcal{H}[M] = (t_M, x_M)$, is the origin of the latest model $\vec{s}_M$, with slope g_M. $\vec{s}_M$ can be used to read past $\mathcal{H}[M]$ and into the future. Gradients $g_{\min}$ and $g_{\max}$ bound the next sample insertions so as to keep their modelling error below ϵ. The three slopes are updated upon insertion. The last inserted point p_{last} is also saved for next iterations. 25

Figure 3.2 When a new sample fits within $[g_{\min}, g_{\max}]$, it is added to the current model by updating $g_{\min}$ and $g_{\max}$. g_M is also updated for read queries (see Alg. 1). 28

Figure 3.3 When an error $> \epsilon$ is reported, a new model is created using p_{last} as its origin. 28

Figure 3.4 CDF of latitude and longitude deltas (in degrees) of successive location samples in Cabspotting and PrivaMov datasets. One can observe that, from one location sample to the next, latitude or longitude deviations are small. . . . 30

Figure 3.5 Compressed storage cost after inserting $1M$ samples, random (R) or constant (C), in SQLite and FLI. 34

Figure 3.6 Memory gain distribution when storing Cabspotting with FLI. 34

Figure 3.7 Throughput for insertions and reads using FLI, SWAB, and Greycat (log scale). FLI drastically outperforms its competitors for insertions and reads. 35

Figure 3.8 Cumulative distribution functions depicting the stability of the FLI linear fragments on PrivaMov & Cabspotting with $\epsilon = 10^{-3}$. 36

Figure 3.9 Model size and gain of FLI storing accelerometer values ($\epsilon = 1$). FLI reports a constant memory whenever stationary, and a small gain ($>\times 1.39$) when walking. 38
Figure 3.10 Demonstration of *univariate* FLI [180]. The original time series $\mathcal{P}$ (in grey) is compressed into its FLI representation $\mathcal{H}$ (in solid black): a series of linear interpolations joint by breakpoints (solid black dots). All dropped points (grey crosses) maintain an error with the FLI interpolation below the configuration parameter ϵ: they remain in the light-red region. 39
Figure 3.11 Projection of the thresholds for uni-dimensional and multi-dimensional instances, fit with latitude and longitude toy data. 40
Figure 3.12 To compress a multivariate time series, Fast Linear Interpolation in n-Dimensions (FLInD) requires two main parameters: the mapping of the input dimensions into different compression groups G, and the array of per-dimension error bound $\vec{E}$. 41
Figure 3.13 Toy example showcasing the influence of a series' alignment on FLInD's compression. Vertical, red-dotted lines mark the breakpoints where an interpolation will be recomputed. FLInD works best when breakpoints occur simultaneously across dimensions. 41
Figure 3.14 Identification of series clusters for compression. Using a value threshold when building the graph of shared timestamps ratios allows us to identify cliques, which hence represent series sharing timestamps. 46
Figure 3.15 Read and write throughput (in operations/second) of linear approximation competitors over several datasets, one per line. The first column lists read operations per second; the second, write operations per second. The red dotted line marks the baseline for the raw storage budget. For each technique, a circle marker marks the lossless variant, and a square marker marks a lossy variant configured to output a comparable MSE of ≈ 0.001. Smaller points additionally present intermediate configurations of competitors with a varying error. 49
Figure 3.16 Architectural diagram of our *in-situ* big data processing experiment on a mobile device. . . 54

Figure 3.17 Distances distribution when using Divide&Stay (D&S) on Cabspotting. The distances between the Points Of Interest (POIs) are obtained using D&S and their closest counterparts, obtained with the traditional POI attack. Except for a few extreme values, the values are close: more than 68% are the same and 90% of the POIs are at a distance lower than 22 meters from a "real" one. 56
Figure 3.18 Example of POI of `user 0` from Cabspotting using the standard POI algorithm on raw data (in **blue**) and those obtained on the data modelled by FLI and computed with Divide&Stay (in **green**). The inferred POIs are very close on average, but there are some outliers. 57
Figure 3.19 Average privacy gain against real data using all dataset users. 58
Figure 4.1 Some of Device-to-device (D2D) communication's features strengthen our desired security properties. 74
Figure 4.2 Distribution of the frequency of the need for D2D communication. 61% of the surveyed users need D2D communication at least once a week, while only 12% need it less than once a month. 80
Figure 4.3 Use and satisfaction (Sat.) of the different available technologies. Only users who frequently use the given technology are considered. Bluetooth is the least popular technology for exchanging documents between nearby devices, despite being designed for this purpose. On the other hand, the cloud is the preferred solution, as it is the only one that does not rely on the device's proximity. 82
Figure 4.4 Device-to-device communication process. The *advertiser* (Adv.) offers a service, while the *discoverer* (Disc.) is looking for interactions. The service discovery is performed on a *bootstrap* channel, while the communication is performed on a *data* channel, see Table 4.3. 83
Figure 4.5 The literature lacks a standard abstraction to unify D2D channels in the link layer. 84
Figure 5.1 Evaluating Quality Attributes of TPLs (EQuAT) for comparing their impact on app quality. 90

Figure 5.2 Distribution of quality metrics for each Third Party Library (TPL) under study, grouped by TPL category. The lines in the boxes indicate the minimum value, lower quartile, median, upper quartile, and maximum values. The "◇" symbol represents the average value. 99
Figure 5.3 Pairwise comparison of TPLs and quality metrics. Each cell contains the difference of the medians in %, the magnitude of the difference (in MB, number of permissions, Joules, %, MB, and MB, for Android Package Kit (APK) size, permissions, energy consumption, CPU, memory, and network, respectively), and the magnitude of the effect size (small, medium, or large). The effect size is "NA" when it does not apply (i.e., for APK size and permissions metrics where we only have a measurement for each TPL). Absent values indicate cases where there is no statistically significant difference. For each cell in the matrix, negative differences (green colours) mean that the TPL at the bottom is better than the TPL at the left. On the contrary, positive differences (red colours) mean that the TPL at the bottom is worse than the TPL at the left. Therefore, greener colours have higher differences in favour of the TPL at the bottom and redder colours have higher differences in favour of the TPL at the left. 101
Figure 5.4 Comparison of Predicted vs. measured share of power usage for the assessed component groups. 104
Figure 5.5 The evolution of R^2 according to the size of the dataset. 105

LIST OF TABLES

Table 3.1 Mobile devices used in the experiments. . . . 33
Table 3.2 Storage budget (in MiB) of compressing datasets; best option is put in bold. 47
Table 3.3 Read and write throughput (in operations/second) of storage techniques over several datasets (reads to the left, writes to the right); best option (excluding raw storage baseline) is put in bold. 48
Table 3.4 Performances metrics over compressing the MICROSERVICES TRACES dataset using an increasingly smaller count of dimensions per instance. The lower the dimensions count, the better the error is acknowledged, leading to better total size, but also to increased read/write times due to the dataset being split between many FLInD instances. 50
Table 3.5 Execution time for PRIVAMOV `user 1` on different mobile platforms. Previous deployments of these algorithms were made on desktop computers (as reference, POI-ATTACK execution takes one hour to complete on a laptop). FLI now enables its instantiation on ubiquitous devices, with reasonable processing times. 56
Table 3.6 Impact of FLI and Divide&Stay on the number of inferred POIs from `user 0` trace in CABSPOTTING. Thanks to FLI and Divide&Stay (D&S), PROMESSE protects user privacy at the edge. . 57
Table 4.1 Characteristics of the most widely available D2D communication channels 67
Table 4.2 Alternate solutions used by participants of our survey for close file sharing, proposed in the free-text field (66 participants left an answer). Technologies are not exclusive, e.g., *Samba* is generally a *Windows Local net* solution. 81
Table 4.3 Preferred communication type per channel . . 85
Table 5.1 Popular Android TPLs for popular TPL categories. 96
Table 5.2 Rankings provided by EQuAT for each quality metric and TPL. 100
Table 5.3 Android permissions and their protection level for the TPLs under study. 102

Table 5.4 The monitored metrics and their Pearson Correlation Coefficient to the component they represent (All correlations are significant, $p < 0.01$). 103
Table 5.5 Accuracy of per-component predictions (all correlations are significant, $p < 0,01$). 104
Table A.1 Results of computing every modelling combination of splitting series between one FLInD instance and one FLInD instance per model, sorted by ascending size. The most optimal way to store the five series is to store all CPU series within one FLInD instance, and model the *GPU Memory* series separately. 141

ACRONYMS

ADB Android Debug Bridge
APK Android Package Kit hello there
AI Artificial intelligence
API Application Programming Interface
AWDL Apple Wireless Direct Link
BLE Bluetooth Low Energy
D2D Device-to-device
D&S Divide&Stay
DTNs Delay-Tolerant Networks
EQuAT Evaluating Quality Attributes of TPLs
ICT Information and Communications Technology
IoT Internet of Things
IrDA Infrared Data Association
IRK Identity Resolution Key
FLI Fast Linear Interpolation
FLInD Fast Linear Interpolation in n-Dimensions
GDPR General Data Protection Regulation
GNC Google Nearby Connections
GPS Global Positioning System
GUI Graphical User Interface
ICDCS International Conference on Distributed Computing Systems
INTACT IN-siTu locAtion proteCTion
IT Information Technology
LBS Location-based service
LLM Large Language Model
LPPM Location Privacy Protection Mechanism
LVPM Low Voltage Power Monitor
MANETs Mobile Ad hoc Networks
MAC Medium Access Control
M2M Machine-to-machine
MPC Model Predictive Control
MTS Multivariate Time Series

NFC Near-Field Communication
ODPM On Device Power Rails Monitor
OppNets Opportunistic Networks
PLA Piecewise Linear Approximation
POI Point Of Interest
QR Quick-Response
RF Radio Frequency
RFID Radio-Frequency IDentification
SDK Software Development Kit
SoC System on a Chip
SPI Sensitive Personal Information
TPL Third Party Library
TPLUT TPL under test
TSDB Time series database
UTS Univariate Time Series
UWB Ultra-wideband
WEC Wi-Fi Easy Connect
WPS Wi-Fi Protected Setup

1

INTRODUCTION

1.1 MOTIVATION

1.1.1 *Context*

Human impact on Earth.

Human activities have been harming the planet since the Industrial Revolution, with some visible indicators.

In 2026, CO_2 levels are 150% the value they were at in 1750 [152], directly contributing to the global warming: the global annual temperature has increased in total by a little more than 1°C since 1850. And despite the 2015 Paris Agreement on limiting the global warming increase to 1.5°C, insufficient mitigation and adaptation action makes it likely that this rise will exceed 2°C [101].

The **Holocene extinction continues**, with the demise of 46% of trees since the advent of human civilisation [44] and an average 73% decline in wildlife populations since 1970 [228].

Since they are also part of the Earth ecosystem, human impacts on the planet also **affect humans themselves**.

The air pollution inherent to human activities, emission of particulate matter (PM) more specifically, is responsible for over 4 million deaths each year, with an increased impact on low and middle-income countries [227].

As a universal solvent, water is particularly vulnerable to pollution; consequently, water pollution kills: in 2015, 1.8 million died from diseases related to water pollution [112].

An additional 500,000 people die each year due to extreme heat, with this count growing exponentially due to climate change [171].

We're at death's door, another world war
Wildfires and earthquakes I foresaw
A life in crisis, mutated virus
Tsunamis of hate are gonna drown us [...]
We are fucking fucked

— Muse

Involvement of ICT.

Throughout their evolution, humans have invented tools to improve their quality of life: agriculture secured their food supply, enabling them to move from a nomadic to a sedentary existence; the steam

[152] *NASA (2026),* Carbon Dioxide - Earth Indicator

[101] *Intergovernmental Panel on Climate Change (2023),* Synthesis report: Climate Change

[228] *World Wide Fund for Nature (2024),* Living Planet Report

[227] *World Health Organization (2025),* Air quality, energy and health

engine revolutionised the way they work and travel, enabling the rise of steam locomotives, assembly-line work, and increased productivity.

In the same fashion, Information Technology (IT) deeply changed the way humans communicate, speeding business processes by deprecating paper through the digitalisation of services and improving contact time through interconnected networks.

Despite its benefits to society, IT **also involves drawbacks**, including cybercrime, personal data trafficking and the segmentation of communities by social networks algorithms.

Additionally, lots of *minerals* are required to build computer, server, or smartphone hardware (water, energy, minerals); some of these resources are increasingly hard to obtain, for both political (*e.g.*, countries outside European Union) and physical reasons (*e.g.*, gold, sparse resources being increasingly harder to extract from the ground).

In opposition, new uses, such as building data centres and electrifying the global car fleet, require increasing resources, which simply is not possible [231].

[231] Yergin et al. (2026), Copper in the Age of AI: Challenges of Electrification

> *The cumulative amount of metals to be produced over the next 35 years would exceed the cumulative amount produced from ancient times to the present day.*
>
> — Aurore Stéphant [198]

[151] Murugesan (2008), "Harnessing Green IT: Principles and Practices"

Acknowledging IT's contribution to global warming, *Green IT* [151] has been proposed to reduce IT impact on the planet, issuing recommendations such as reducing the energy consumption of computers, reusing and recycling old hardware, and designing energy-efficient components. However, Green IT's optimistic rhetoric seems insufficient and is challenged at the time of writing [111].

[111] Knowles et al. (2025), "Climate Change: What is Computing's Responsibility? (Dagstuhl Perspectives Workshop 25122)"

Cloud, integral part of ICT impact.

Despite the initially decentralized configuration of the Internet, services are nowadays structured in a *centralized* way, where users employ terminal devices (computers, smartphones) to access services running on **servers**, conveniently regrouped in **data centres**.

The cloud paradigm allowed for resource mutualization, scalability, flexibility and cost savings for companies, to the cost of vendor lock-in and control loss over server infrastructure [70].

[70] Google (2019), The Future of cloud computing

Besides, the cloud impact on the planet is **far from being negligible**, contrasting with companies that claim to be "carbon-negative", when in fact they are merely compensating the ever-growing pollution they cause [138].

[138] Microsoft (2025), Environmental Sustainability Report

All cloud-hosted services do not contribute equally to the cloud impact on the planet.

As an edifying example, Artificial intelligence (AI) is a trending technology at the time of writing, every business integrating Large Language Models (LLMs) to their services in fear of being left behind in this "AI race": for instance, Meta forcibly (with no way of disabling it) integrated its Meta AI model in Messenger and WhatsApp chatting applications.

The AI race even leads some businesses to trade privacy for AI performance in their strategic decisions, *e.g.*, Google firing employees to accelerate the development and release of AI-related products [166] or some to lie about their activities to get funding, *e.g.*, builder.ai claiming to use AI to generate websites while they were secretly employing human workers [194].

[166] Politico (2024), Google makes changes to privacy oversight, worrying policymakers

To conclude, the AI/cloud combination has a **non-negligible impact on the people**: in the U.S. in 2023, the public health cost of data centre-related pollution was 168.81$ per household (25% of the annual health budget) [81]. In 2023, AI was also thirsty [117], with Google's data centres alone consuming 29 billion litres of water for on site cooling; 80% of this water was potable, leading to conflicts with local population [57].

[81] Han et al. (2025), The Unpaid Toll: Quantifying and Addressing the Public Health Impact of Data Centers

[57] Fleury et al. (2025), 'I can't drink the water' - life next to a US data centre

1.1.2 *Problem statement*

End users use the cloud every day, whether by watching series on Netflix, listening to music on Spotify, or attending meetings remotely via Zoom. Not knowing servers run in data centres far away from their homes, users also do not acknowledge that the devices they use to access services, be it smartphones, desktop computers, or even smart TVs, are made of the same elements as servers (namely CPU, memory, and storage units), and as such also have computing power.

Local computing, as opposed to cloud computing, brings in several benefits: (i) *security guarantees*, as the use of the Internet is not required, (ii) *privacy protection*, as trusting a cloud provider with personal information has proven subject to privacy intrusions [61], and (iii) *environmental friendliness*, as eschewing any dependence on the Internet lowers the pressure on its infrastructures.

[61] Gegenhuber et al. (2026), "Hey there! You are using WhatsApp: Enumerating Three Billion Accounts for Security and Privacy"

However, local computing also faces constrained resources, including low CPU performance, limited network connectivity, and degraded battery performance.

To protect end-users from data-hungry big tech companies, we want to bring data processing from cloud providers as close as possible to end-user devices, if not directly on end-users' smartphones. As such, taking an adversarial stance toward the cloud, this thesis explores how the latter can be bypassed by providing services locally, ensuring individual privacy protection and local communication. This being said, the cloud offers some advantages, reduced energy consumption due to shared resources being one of them. This thesis also examines

how the cloud can be used to respect end users' privacy, using location privacy as a use case.

Additionally, as reduced energy consumption of the cloud compared to smartphones is one argument Google uses to recommend relying on cloud computing [162], the thesis examines smartphone energy consumption and related applications to better understand it and propose concrete actions to reduce it.

[162] Patterson et al. (2024), "Energy and Emissions of Machine Learning on Smartphones vs. the Cloud"

1.2 CONTRIBUTIONS

This thesis revolves around three distinct domains, sequentially exploring *(i)* data local storage and computing, *(ii)* local communication, and *(iii)* energy usage, all of which are applied to smartphones.

First, delving into the intersection of local storage and user privacy, the first contributions explore how simple linear interpolation algorithms can be used to drastically improve the storage capacity of mobile devices, using location data as a use case. The second, extended version demonstrates how these algorithms enable *in situ* computing, enabling location-based attacks on user data to run locally and thereby improving user privacy. The third contribution extends previous work on Univariate Time Series (UTS) and applies it to Multivariate Time Series (MTS), addressing challenges arising with high-cardinality series.

The second domain explores the risks of using the cloud and the Internet more globally and, in two subsequent contributions, *(i)* demonstrates that local device-to-device communication is 1) requested by both end users and application developers and 2) underused, and *(ii)* explores a high-level, tech-abstract communication framework and its implementation.

Last but not least, taking the energy consumption of the mobile ecosystem into account, and in order to identify the places where it can be improved, the last two contributions *(i)* identify the phone components responsible for the majority of the energy usage and *(ii)* help application developers choose the most efficient libraries to use in their projects, to reduce the overall consumption of applications.

1.2.1 *Protecting user privacy through local data storage*

Location data, as Sensitive Personal Information (SPI), is valuable to third parties, as it can reveal a person's place of work or residence. However, if a user wants to benefit from a Location-based service (LBS), she needs to send her data to the LBS provider, putting her at risk.

To address this problem, we propose FLI, a linear regression-based algorithm that allows storing gigabytes of location and UTS data on mobile phones.
Secondly, we present FLInD, a multi-dimensional generalisation of FLI,

that extends it to support series with multiple dimensions (MTS).
Then, we integrate previous compression algorithms into the IN-siTu locAtion proteCTion (INTACT) framework, a tool that runs entirely on smartphones and supports local data attacks and subsequent data transformations, to detect and fix SPI leaks before sharing data with peers.
Finally, we study how linear regression can be leveraged to detect and fix privacy leakages *before* they occur.

1.2.2 *Unlocking local device-to-device communication*

Nowadays, services providing relies on the cloud paradigm, with servers running in data centres: this raises issues such as compromised privacy and environmental impact, while widespread Device-to-device (D2D) technologies such as Bluetooth theoretically allow to remove this dependency to the cloud.

The first contribution in this domain is a survey that explores the history of D2D communication, the reasons for its unpopularity, and the need for a non-Internet, unified data-link communication layer.
This subsequent contribution presents *Venice*, a technology-agnostic framework that leverages locally available communication channels to exchange data between devices in a face-to-face manner.

1.2.3 *Improving the energy consumption of mobile devices*

Previous studies showed that TPLs accounts for two-thirds of method calls in applications. To help developers better manage app consumption, in our first contribution, we propose EQuAT, an empirical approach that analyses various metrics from real deployments to compare TPLs and select the best for a given use case.
Finally, we propose a robust approach to model smartphone energy consumption using hardware sensors, enabling developers to identify the source of energy drain.

1.3 LIST OF SCIENTIFIC PUBLICATIONS

1.3.1 *Published*

Parts of this thesis are adapted from the following publications:

1. Rémy Raes et al. "Venice: eschewing the cloud by leveraging local communication channels." In: *Joint Proceedings of ICT4S 2024 Doctoral Symposium, Demonstrations & Posters Track and Workshops*. Stockholm, Sweden, June 2024, pp. 1–4. URL: https://hal.science/hal-04576743

2. Rémy Raes et al. "Compact Storage of Data Streams in Mobile Devices." In: *Proceedings of the 24th International Conference on Distributed Applications and Interoperable Systems (DAIS'24)*. LNCS. Groningen, Netherlands: Springer, June 2024. DOI: 10.1007/978-3-031-62638-8_4. URL: https://hal.science/hal-04535716

3. Emilio Molina et al. "Application of a predictive method to protect privacy of mobility data." In: *Control Engineering Practice* 156 (Mar. 2025), p. 106223. DOI: 10.1016/j.conengprac.2024.106223. URL: https://hal.science/hal-04885266

4. Édouard Guégain et al. "AndroWatts: Unpacking the Power Consumption of Mobile Device's Components." In: *Proceedings of the 12th International Conference on Mobile Software Engineering and Systems (MOBILESoft'25)*. Ottawa, Canada, Apr. 2025. DOI: 10.1109/MOBILESoft66462.2025.00015. URL: https://hal.science/hal-04928609

5. Lauric Desauw et al. "A Critical Review of Mobile Device-to-Device Communication." In: *Proceedings of the 25th International Conference on Distributed Applications and Interoperable Systems (DAIS'25)*. Ed. by Daniel Balouek et al. LNCS. Springer, June 2025, pp. 1–24. ISBN: 978-3-031-95728-4. DOI: 10.1007/978-3-031-95728-4_1. URL: https://inria.hal.science/hal-04198528v1

6. Rémy Raes et al. "INTACT: Compact Storage of Data Streams in Mobile Devices to Unlock User Privacy at the Edge." In: *Journal of Internet Services and Applications (JISA)* 16.1 (June 2025), 372–387. DOI: 10.5753/jisa.2025.5242. URL: https://journals-sol.sbc.org.br/index.php/jisa/article/view/5242

7. Rubén Saborido et al. "Simply the Best -A Systematic Evaluation Approach for Third-Party Libraries based on Mobile App Quality Attributes." In: *Empirical Software Engineering* 31.5 (May 2026), p. 137. DOI: 10.1007/s10664-026-10872-w. URL: https://link.springer.com/article/10.1007/s10664-026-10872-w

1.3.2 *Under submission*

Additional contributions are under submission:

1. Less is More: a Lossy Storage Middleware for Multivariate Time Series
   (submitted to International Conference on Distributed Computing Systems (ICDCS))

1.4 PROOFS OF CONCEPT

This thesis has resulted in the creation of a tool and multiple prototypes. The complete source code for all artefacts has been made available. The list of artefacts is as follows:

- FLI + INTACT
  Applications: GitLab (preserved on Software Heritage)
- FLIND
  Notebook: Software Heritage
- VENICE
  Applications: GitHub (preserved on Software Heritage)
- ANDROWATTS
  Android application: Gitlab (preserved on Software Heritage)
  Metrics extraction: Github (preserved on Software Heritage)
  Traces analysis: Zenodo
- ANDROID TPLS
  Benchmark: GitHub (preserved on Software Heritage)

1.5 OUTLINE

This thesis is structured as follows. First, I present the background and related work to this thesis in Chapter 2. Then, I discuss my contribution in three distinct domains: in Chapter 3, I present how leveraging simple regression formulas allows for increased privacy by introducing local data storage and attack layers on mobile; in Chapter 4, I advocate for global adoption of the Device-to-device (D2D) paradigm, and propose a framework to that end; I then discuss in Chapter 5 how developers can identify opportunities to improve the energy efficiency of their applications. Lastly, Chapter 6 concludes this thesis by summarising the contributions and proposing research perspectives.

2

BACKGROUND

Since the introduction of the first iPhone in 2007, smartphones have changed the way we consume and generate information, leading to major economic and societal changes, including, for instance, the rise of "Big Tech" companies and the shift of traditional media to social networks.

In this context, this chapter explores the general question of control and autonomy left to users, notably in terms of privacy (and its implications on both data storage and communication) and energy efficiency. In this chapter, we focus on three key questions linked to this broad issue, related to the contributions we present in the remainder of this manuscript:

(1) Storage and protection of data produced by our smartphones (Section 2.1);

(2) Bypassing the centralized intermediate actors through direct terminal-to-terminal communication (Section 2.2);

(3) The impact of hardware and software choices on the energy consumption of smartphones (Section 2.3).

Contents

2.1 Protecting user privacy through local data storage **10**
2.1.1 Location Privacy Attacks 10
2.1.2 Protecting Mobility Datasets at once 10
2.1.3 Protecting Individual Traces 10
2.1.4 Data Storage 11
2.2 Unlocking local device-to-device communication . **13**
2.2.1 Introduction 13
2.2.2 Motivation 14
2.3 Understanding and improving mobile energy usage **17**
2.3.1 Usage of Third Party Libraries (TPLs) 17
2.3.2 Impact of Third Party Libraries (TPLs) . . . 18
2.3.3 Analysis of Implementation and Design Decisions 19
2.3.4 TPL Ecosystem: Identification, Selection, and Quality 20
2.4 Summary **21**

2.1 PROTECTING USER PRIVACY THROUGH LOCAL DATA STORAGE

In this Section, we cover the topics of mobility data, existing location privacy attacks, and related protection mechanisms, including Location Privacy Protection Mechanisms (LPPMs), before focusing on state-of-the-art databases used to store data streams.

2.1.1 *Location Privacy Attacks*

Raw user mobility traces can be exploited to model users' behaviour and reveal Sensitive Personal Information (SPI). In particular, Points Of Interest (POIs) are widely used as a way to extract SPI from mobility traces. In a nutshell, a POI is a place where the user often comes and stays for a significant amount of time: it can reveal their home, workplace, or leisure habits. From revealed POIs, more subtle information can also be inferred: sexual orientation from attendance to LGBT+ places, for instance. The set of POIs can also be used to re-identify a user in a dataset of mobility traces [60, 169]. The POIs can be extracted using spatio-temporal clustering algorithms [82, 234]. Alternatively, an attacker may also re-identify a user directly from raw traces, without computing any POI [131].

[169] Primault et al. (2014), "Differentially private location privacy in practice"

[60] Gambs et al. (2014), "De-anonymization attack on geolocated data"

[131] Maouche et al. (2017), "Ap-attack: a novel user re-identification attack on mobility datasets"

2.1.2 *Protecting Mobility Datasets at once*

When data samples are gathered in a remote server, one can expect the latter to protect the dataset as a whole. In particular, *k-anonymity* [200] is the property of a dataset guaranteeing that whenever some data leaks, the owner of each data trace is indistinguishable from at least $k-1$ other users contributing to the dataset. Similarly, *l-diversity* [127] extends *k-anonymity* by ensuring that the l users are diverse enough not to infer SPI about the data owner. Finally, *differential privacy* [53] aims to ensure that including a single element in a dataset does not significantly alter an aggregated query over the entire dataset. However, all these techniques require personal samples to be grouped to enforce user privacy.

[200] Sweeney (2002), "k-anonymity: A model for protecting privacy"

[127] Machanavajjhala et al. (2007), "l-Diversity: Privacy beyond k-anonymity"

2.1.3 *Protecting Individual Traces*

Rather than protecting the dataset as a whole, each data sample can also be protected individually. For location data, several protection mechanisms—called LPPMs—have been developed. They may be deployed in a remote server where all data samples are gathered, or run directly on the source device before any data exchange, provided there is sufficient storage space.

[53] Dwork (2008), "Differential privacy: A survey of results"

GEO-INDISTINGUISHABILITY (GEOI) [10] implements differential privacy [53] at the trace granularity. In particular, GEOI adjusts mobility traces with two-dimensional Laplacian noise, making POIs more difficult to infer. *Heat Map Confusion* (HMC) [130] aims to prevent re-identification attacks by altering all traces. The raw traces are transformed into heat maps, which are then altered to match another heat map in the dataset, and then transformed back into a POI trace.

[10] Andrés et al. (2013), "Geo-indistinguishability: Differential privacy for location-based systems"

PROMESSE [170] smooths the mobility traces, both temporally and geographically, to erase POIs from the trace. PROMESSE ensures that, between each location sample, there is at least a given time and distance interval. In the resulting mobility trace, the user appears to have a constant speed. While PROMESSE blurs the time notion from the trace—*i.e.*, the user never appears to stay at the same place—it does not alter their spatial characteristics. Yet, while POIs may still be inferred if the user repeatedly goes to the same places, it will be harder to distinguish such POIs from more random crossing points.

[170] Primault et al. (2015), "Time distortion anonymization for the publication of mobility data with high utility"

It is also possible to combine several LPPMs to improve user privacy [109, 135]. Because of potential remote leaks, the user should anonymise their trace locally before sharing it, as EDEN [109] does. However, EDEN has not been deployed: it has only been simulated on the Android Debug Bridge (ADB).

Even more so: despite their validity and to the best of our knowledge, no LPPM has been implemented on mobile devices. This is partly due to the tight constraints of mobile devices, notably memory-wise: HMC [130], for instance, requires loading a large set of Global Positioning System (GPS) traces locally to operate.

2.1.4 *Data Storage*

Storing data streams on mobile devices is difficult, as one must overcome tight memory constraints, which requires efficient techniques specifically tailored for temporal data. Unfortunately, few databases are available for constrained devices. For instance, Android developers must choose between SQLite and its derivative Drift [25], the cloud-supported Firebase [202], the NoSQL database Hive, and ObjectBox [50]. The situation is similar on iOS.

In the following, we review three broad families of databases (Relational databases, moving objects databases, and Data stream storage) and discuss to what extent they fit the needs of temporal data storage on mobile devices.

Relational databases (*e.g.*, SQL) are typically designed for *OnLine Transactional Processing* (OLTP) and *OnLine Analytical Processing* (OLAP) workloads, which widely differ from time-series workloads. In OLAP, reads are mostly contiguous, as opposed to the random-read tendency of OLTP; with time-series workloads, writes are most often inserts

Here, "online" describes the behaviour of a system processing data as it arrives, in real time.

(not updates) and typically target the most recent time ranges. OLAP is designed to store large data workloads for analytical computation, without emphasising read or write performance. Finally, in temporal workloads, it is unlikely to process writes & reads in the same single transaction [205].

[205] Timescale (2019), Building a Distributed Time-Series Database on PostgreSQL

Time series databases (TSDBs). Despite these big differences, several relational databases offer support for temporal data with industry-ready performance—*e.g.*, TimescaleDB [204] is a middleware that exposes temporal functionalities atop a relational PostgreSQL foundation. InfluxDB [99] is one of the most widely used temporal databases. Unfortunately, when facing memory constraints, its retention policy prevents storage from scaling over time: the oldest samples are discarded to make room for the new ones. Furthermore, on mobile devices, memory shortages often cause the operating system to kill the TSDB process to free memory, which contradicts the very concept of in-memory databases.

Moving objects databases (MOD). Location data storage is also an issue that has been studied in the MOD community, where a central authority merges trajectory data from multiple sensors in real time. To optimise storage and communication costs, it does not store the raw location data, but rather trajectory approximations. *Linear Dead Reckoning* (LDR) [226] limits data exchange between sensors and server by sending new location samples only when a predefined *accuracy bound* ϵ (in meters) is exceeded. A mobility prediction vector is also shared whenever a location sample is sent. Even so, this class of solutions requires temporarily storing locations to ensure they meet the ϵ bound and focuses exclusively on modelling location data streams, whereas we aim to store any type of real-valued stream.

[226] Wolfson et al. (1998), "Cost and imprecision in modeling the position of moving objects"

Modelling data streams. While being discrete, the streams sampled by sensors represent inherently continuous signals. Data modelling not only enables significant memory savings but also reduces sensor noise and enables extrapolation between measurements. In particular, Piecewise Linear Approximation (PLA) is used to model the data as successive affine functions. An intuitive way to do linear approximation is to apply a bottom-up segmentation: each pair of consecutive points is connected by interpolation; the less significant contiguous interpolations are merged, as long as the obtained interpolations introduce no error above a given threshold.

The bottom-up approach is low-complexity but usually requires an offline approach to consider all points at once. The *Sliding Window And Bottom-up* (SWAB) algorithm [108], however, is an online approach that uses a sliding window to buffer the latest samples on which a bottom-up approach is applied. emSWAB [23] improves the sliding window by adding several samples at the same time instead of one. Instead of interpolation, linear regression can also be used to model

[108] Keogh et al. (2001), "An online algorithm for segmenting time series"

[23] Berlin et al. (2010), "An on-line piecewise linear approximation technique for wireless sensor networks"

the data from Internet of Things (IoT) sensors [78]. For example, GREYCAT [142] adopts polynomial regressions with higher degrees to further compress the data. Unfortunately, none of those works have been implemented on mobile devices to date.

SPRINTZ [26] proposes a mobile lossless compression scheme for multi-modal integer data streams, along with a comparison of other compression algorithms. They target streaming of the compressed data to a centralised location from IoT devices with minimal resources.

Rather than streaming data to a third-party server, floating-point unimodal streams can be captured on one's devices for local computation. FSW [120] and the SHRINKINGCONE algorithm [59] attempt to maximise the length of a segment while satisfying a given error threshold. FSW is not a streaming algorithm because it considers the dataset as a whole and does not support insertions. The SHRINKINGCONE algorithm is a streaming, greedy algorithm designed to approximate an index that maps keys to positions; it considers only monotonic, increasing functions and may produce disjoint segments.

[142] Moawad et al. (2015), "Beyond discrete modeling: A continuous and efficient model for IoT"

[26] Blalock et al. (2018), "Sprintz"

[120] Liu et al. (2008), "Novel online methods for time series segmentation"

[59] Galakatos et al. (2019), "FITing-tree: A data-aware index structure"

Limitations. To the best of our knowledge, state-of-the-art storage solutions for unbounded data streams either require storing raw data samples or triggering *a posteriori* data computations, which makes them unsuitable for mobile devices.

2.2 UNLOCKING LOCAL DEVICE-TO-DEVICE COMMUNICATION

While local data storage provides privacy guarantees, these are not valid when communicating over networks via intermediate peers (*e.g.*, the Internet). In this Section, we present how the traditional cloud-based communication model is challenged by Device-to-device (D2D) communication.

2.2.1 *Introduction*

Device-to-device (D2D) refers to the capability of mobile devices to communicate directly with one another, avoiding intermediaries, such as a cellular base station. The D2D paradigm has been enabled by a wide range of wireless communication channels (e.g., infrared, Wi-Fi, Bluetooth, Near-Field Communication (NFC), Ultra-wideband (UWB)) for a long time, but it has yet to be widely adopted in user-facing applications. It is currently hindered by technical (and non-technical) barriers that make it an unsuitable choice for developers and end-users alike.

Most networked applications thus rely on centralised infrastructures, even for inherently local use cases, such as proximity sensing and face-to-face media exchanges—which limits the capacity of pervasive cyber-physical systems to scale [39, 40].

D2D communication would have an important role in modern communication, however, due to its low environmental footprint, inherent privacy, and the pathway it paves to pervasive services beyond the Internet.

2.2.1.1 *Scope*

'Device-to-device communication' is an umbrella term that has many definitions. In this manuscript, we define D2D with a user-centric focus. We consider D2D communication as the exchange of information that occurs *wirelessly*, in a *single hop* (without intermediaries), using channels and protocols already available on commodity *end-user mobile devices*. Intermediaries, such as a cellular base station, might assist in establishing the connection but should not participate beyond that.

This manuscript does not cover Machine-to-machine (M2M) applications, although it does mention some M2M-oriented protocols. Indeed, automatic communication between machines is not user-centric and generally does not leverage the same protocols as those available on end-user equipment.

2.2.1.2 *D2D scenarii*

[105] Jameel et al. (2018), "A Survey of Device-to-Device Communications"

Jameel *et al.* [105] proposed a taxonomy of D2D use cases, namely traffic offloading, providing emergency services, extending cellular coverage, reliable health monitoring, mobile tracking and positioning, and data dissemination. Among these use cases, we do not address provisioning emergency services, as it relies on multi-hop communication via advanced routing. Health monitoring and mobile tracking are left aside because they rely on specialised sensor networks outside the end users' realm (e.g., M2M communication). Traffic offloading and the extension of cellular coverage focus on facilitating access to Internet services and are thus not our focus, although they will be mentioned.

Our work specifically targets *user-centric data dissemination*, ranging from direct file sharing to collaborative social services such as crowdsourced environmental sensing. An example of the file-sharing scenario is shown in Figure 2.1, where, in contrast to using Internet services, two users who share a file directly achieve better energy efficiency, lower resource consumption, and better privacy. Our online survey, presented in sec. 4.1.4, shows that participants encounter this use case regularly.

2.2.2 *Motivation*

Networked user applications are thriving, and Internet traffic is growing at an astonishing rate [37, 225]. Yet, nearly all users' network sessions transit through the *cloud*: a collection of data centres equipped with high-grade equipment and physically connected as close as possi-

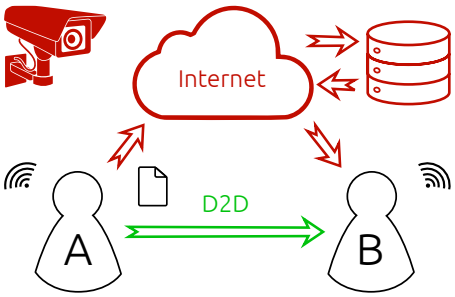


Figure 2.1: For an inherently D2D use case such as transferring a file between nearby users, resorting to an Internet service for the exchange leads to more energy consumption, more storage space used, and exposes the users to digital surveillance (degraded privacy).

ble to the Internet's 'backbone'. In this Subsection, we develop several arguments in favour of greater D2D communication in mobile device networking.

2.2.2.1 *Supporting the energy transitions*

We first consider the example depicted in Figure 2.1, where Alice and Bob, who are physically close, want to share a file. Unless both users own Apple products that feature AirDrop, as of now, the typical 'netizen' will adopt an indirect solution by uploading files to an online service (e.g., file storage, e-mail, instant communication) before their recipient can fetch them. The alternative is more direct and leverages a D2D protocol (e.g., Bluetooth or Wi-Fi Direct), so that the file is transmitted immediately from Alice's smartphone to Bob's, without the intervention of any intermediaries.

Netizen: *Person who regularly uses the Internet.*

How is the indirect solution more costly? To answer, we need to break down an online-based file exchange: Alice's file needs to be transmitted from her smartphone to her network gateway (either her Wi-Fi access point or her cellular base station), and then over the Internet to the data centre, where the online service will store the file on persistent storage. Then, Bob needs to connect to the same service, so the file follows a similar journey back from the data centre's storage, over the Internet, to Bob's network gateway, and then to Bob's smartphone.

This *sharing-as-a-service* process supported by the cloud is costly in several regards, including increased load on all network equipment between users [167] and increased load on storage, third-party servers storing a copy of the file that could remain there for an unknown time [36].

[167] Price (2017), "Google Drive Now Hosts More than 2 Trillion Files"

[36] Cheng (2012), "Over 3 Years Later, "Deleted" Facebook Photos Are Still Online"

By shifting direct communication between user devices, the aforementioned costs would be highly limited. Only the incompressible energy and storage required to locally transfer the file between the two devices would be consumed. In essence, to lower the Internet's environmental impact, it is crucial to shorten communication chains

wherever possible. We postulate that—given the digitalisation of our lives and the variety of use cases that could avoid relying on a third-party cloud—the raw material savings enabled by D2D communication are substantial, thanks to its lower electrical consumption and negligible infrastructure use.

2.2.2.2 *Unplugging the surveillance economy*

Every time Alice makes a connection or *request* to a data centre to use an online service, the following actors obtain details of the communication:

- The network operators (notably Alice's *Internet Service Provider* (ISP) or cellular network provider), as well as the data centre's owner, obtain Alice's IP address, the potentially encrypted payload, its size, and the request timestamp. If the connection is not encrypted, they can also read the request's content.

  Even when encryption is enforced, a more intrusive operator may also be able to obtain information about Alice's device or to inspect the content of her requests—for example, through *Deep Packet Inspection* (DPI) [18];

[18] Asghari et al. (2013), Deep Packet Inspection

- In addition to the above, the cloud service operator obtains application-specific information, as well as the payload of the request. Following our file-sharing use case, the cloud service obtains the filename, size, content, and, likely, the recipient's address or identifier (e.g., their e-mail address).

Network service providers are legally bound to retain this data for security purposes [43]. Besides this legal requirement, Shoshana Zuboff [236] describes how the digital economy revolves around the accumulation and processing of personal data for financial benefit. Although each datum is seemingly harmless, this "surveillance economy" poses serious threats to individuals [7], free speech [154], and democratic societies at large [33, 123, 133]. By rejecting third parties, D2D communication holds great potential to prevent the accumulation and marketisation of personal information and its detrimental effects.

[43] Court of Justice of the European Union (2020), Judgment on Traffic Log Retention

[236] Zuboff (2019), The Age of Surveillance Capitalism

2.2.2.3 *Uneven access to a broadband Internet connection*

Worldwide access to the Internet and to mobile devices is highly uneven: in 2021, almost half of the World's population was kept offline, according to the United Nations [211], despite 94% of people being within coverage of a mobile broadband network [58]. The geographical distribution of people without access to the Internet shows that Africa is the continent of most concern [102], but there are local disparities within many countries [215].

People's budgets may be the main cause of this digital divide. Indeed, users need to buy a device and, arguably more importantly,

[211] United Nations (2021), With Almost Half of World's Population Still Offline, Digital Divide Risks Becoming 'New Face of Inequality', Deputy Secretary-General Warns General Assembly

they must pay service fees to access the Internet. Prices vary widely per country and depend on many factors. And, while unlimited data plans do exist, most people still pay per gigabyte consumed or have limited plans. Additionally, as data usage increases due to growing volumes of content (such as video streaming), connection fees can be a burden for many, including in developed countries like the U.S. [134].

[134] Mcclain (2021), "34% of Lower-Income Home Broadband Users Have Had Trouble Paying for Their Service amid COVID-19"

By enabling D2D communication without fees, D2D communication would help reduce those costs and contribute to the eradication of the digital divide. Local communications can also provide basic connectivity when Internet coverage is unavailable or inconsistent (e.g., on planes, inside buildings, or during disasters).

2.3 UNDERSTANDING AND IMPROVING MOBILE ENERGY USAGE

Local data storage and local communication both provide privacy guarantees to users; developers must, however, ensure these guarantees do not come at the expense of increased battery drain. This Section thus explores the impact of software on smartphone energy consumption, specifically the impact of Third Party Libraries (TPLs) on Android smartphone energy consumption.

2.3.1 *Usage of Third Party Libraries (TPLs)*

The pace of smartphone growth around the world is unprecedented, helping billions of new app users come online for the first time in what Google calls *build[ing] for the next billion* [74]. Thus, the use of the Android operating system continues to grow. There are now over three billion monthly active Android devices [20]. More than 2 million apps were available on the Google Play Store in 2025[3], and they were downloaded 113 billion times by people from 190 countries, generating $47 billion in revenue [45].

[74] Google (2024), Build for Billions

[20] Backlinko (2025), iPhone vs Android Statistics

[3] 42matters (2025), Google Play vs the Apple App Store: App Stats and Trends

Users enjoy mobile apps that meet their quality expectations, but excessive battery, memory, and network usage, slow render times, crashes, large sizes, and excessive permissions are sources of frustration for them [6]: (1) larger apps take longer to download/install and take undue storage, (2) the numbers and types of requested user permissions affect download and retention [68], (3) energy consumption is critical because mobile devices depend on their batteries to work, (4-5) CPU and memory limit the number of apps users can run, and (6) network access may be expensive.

[6] Ahn (2017), How we're helping people find quality apps and games on Google Play

[68] Google (2018), App permissions best practices

In many countries, this directly conflicts with constraints such as devices with low memory and poor processor performance, or limited opportunities to recharge batteries during the day [74]. To address these users' needs, resources demanded by apps must fulfil user and–or device constraints.

Technology companies care about users' constraints, especially in developing countries, releasing "light" versions of their apps. Facebook released Facebook Lite, an app designed to work on all networks, even slow or unstable connections. Facebook Lite is less than 2MB, so it downloads quickly and uses less storage space. It has been downloaded more than 1 billion times. Other examples include TikTok Lite and Android Go [75].

[75] Google (2026), Android (Go edition)

However, not all app developers have the resources to produce light versions of their apps. On the contrary, they often use Third Party Libraries (TPLs) to speed up development without a clear understanding of the impact of these TPLs on the sizes, permissions, and performances of their apps. They also lack comparisons of the different TPLs to make informed decisions. Yet, the sizes, permissions, and performances of apps correlate with the addition and usage of TPLs. For example, Minelli et al. [140] observed that external calls to TPLs represent more than 75% of the total number of method invocations in apps.

[140] Minelli et al. (2013), "Software Analytics for Mobile Applications-Insights & Lessons Learned"

App developers need information on the impact of TPLs on app quality: app size, the number and types of permissions, and performance.

2.3.2 *Impact of Third Party Libraries (TPLs)*

Previous studies by Wang et al. [216], on more than 100,000 Android apps from five different Android markets, showed that more than 60% of the sub-packages in Android apps are from TPLs. On average, TPLs account for more than 60% of the code in Android apps.

[216] Wang et al. (2015), "WuKong: A Scalable and Accurate Two-phase Approach to Android App Clone Detection"

Other studies showed *(1)* the inconsistencies between apps descriptions and requested permissions due to the usage of TPLs [77, 161], *(2)* the relationship between apps' ratings and TPLs [183], *(3)* the impact of TPLs on app clone detection [216], *(4)* privacy issues due to third-party tracking [159], and *(5)* the inconsistencies between withdrawal decisions and apps' third-party data collection [52].

[183] Ruiz et al. (2014), "Impact of Ad Libraries on Ratings of Android Mobile Apps"

[159] Paci et al. (2023), "A Comprehensive Study on Third-Party User Tracking in Mobile Applications"

Yet, a systematic literature review by Zhan et al. [233] shows that no previous work proposed an approach to measure the impact of TPLs on the quality of apps: developers do not have the comparisons needed to make informed decisions about the TPLs to integrate (or not) in their apps.

Therefore, when choosing a TPL (e.g., for advertising, crash-reporting, or monitoring), developers typically proceed making decisions without a systematic evaluation [113]:

[113] Larios Vargas et al. (2020), "Selecting third-party libraries: the practitioners' perspective"

- Relying on general reputation or popularity. The developer may assume that the most popular library (e.g., Google AdMob for advertising, Firebase for analytics) is the best choice. They may search online ("best advertising Software Development Kit (SDK)

for Android") and pick one from a blog post, Reddit, or Stack Overflow discussion. Without detailed performance testing, they risk choosing a bloated or inefficient library.

- Checking the library's stated performance claims. Developers might consult official documentation or the SDK's website, which often highlights performance benefits but downplays drawbacks (e.g., resource usage and battery drain). If the SDK claims to be "lightweight" or "optimised", they may take it at face value without further testing.
- Looking at APK or AAR file size only. While size is a factor, developers may ignore runtime performance, such as CPU and memory usage, if they only analyse the impact of TPLs on the app's size.
- Running basic profiling. An uninformed developer might not run profiling tools such as the Android Profiler or Battery Historian. If they do test, they may only check one metric (e.g., CPU usage in a short test session) rather than analysing multiple performance aspects over time.
- Testing on a High-End Device Only If the developer tests performance, they might do so only on a flagship device (e.g., Pixel 8, Galaxy S24), where performance issues are less noticeable. They may not test on mid-range or low-end devices, which experience greater performance impacts from heavy TPLs.
- Ignoring background performance impact. Some TPLs consume resources even when not actively used (e.g., advertising TPLs making background network requests, analytics running periodic syncs, etc.). An uninformed developer might only test the library when actively used in the foreground, missing long-term performance degradation.

While previous studies have identified quality-related factors as strong determinants of TPL selection from a practitioner's perspective, there remains a lack of empirical frameworks that provide quantitative comparisons of quality attributes [113], particularly in the mobile context. Thus, TPL selection is usually characterised by a reliance on proxy metrics and qualitative perceptions rather than empirical evidence. This reliance on subjective, ad hoc criteria creates a void.

2.3.3 *Analysis of Implementation and Design Decisions*

Considerable research has examined how developer decisions during the implementation phase determine an app's eventual resource footprint. This includes evaluating the impact of code obfuscation techniques [187], identifying energy-greedy Android APIs [118], and

[187] Sahin et al. (2014), "How Does Code Obfuscation Impact Energy Usage?"

measuring the overhead of specific system-level primitives such as storage subsystems [143] or database transactions [125]. More recently, Saborido et al. [186] provided a fine-grained comparison of primitive data structures (e.g., `ArrayMap` vs. `HashMap`), demonstrating that internal implementation choices can lead to significant CPU and memory savings.

[143] Mohan et al. (2017), "Storage on Your SmartPhone Uses More Energy Than You Think"

Other studies shifted the focus toward automated optimisation and refactoring. For instance, EARMO [149] was proposed to detect and automatically correct energy-intensive anti-patterns within an app's source code to extend battery life. These studies primarily focus on reactive optimisation, refactoring, or modifying internal source code after integration. In contrast, EQuAT enables proactive evaluation by treating TPLs as stand-alone units. By benchmarking TPLs in an isolated, minimal environment, our approach enables developers to compare TPLs across several quality attributes.

[149] Morales et al. (2018), "EARMO: An Energy-Aware Refactoring Approach for Mobile Apps"

2.3.4 *TPL Ecosystem: Identification, Selection, and Quality*

As TPLs dominate a large portion of modern app packages [140, 216], research has intensified on their identification and their broader impact on app quality. Tools such as LibRadar [126], LibD [116], and Libloom [87] address the technical challenge of accurately detecting TPLs in obfuscated or cloned binaries. Other studies have evaluated the external consequences of TPL usage, such as permission inconsistencies [77] or the correlation (or lack thereof) between the number of ad libraries and user ratings [183].

[126] Ma et al. (2016), "LibRadar: Fast and Accurate Detection of Third-party Libraries in Android Apps"

[77] Gorla et al. (2014), "Checking App Behavior Against App Descriptions"

While identification tools are essential for security and compliance, they do not provide a quantitative assessment of a library's runtime behaviour. Furthermore, studies on the performance impact of TPLs have historically been category-specific, focusing almost exclusively on ad networks [80, 185]. EQuAT diverges from this by offering a general-purpose and automated benchmarking framework. We transition from the subjective quality perceptions reported by practitioners [113] to an objective, evidence-based measurement of TPL quality.

Beyond practitioner perspectives on performance, a significant gap persists: the lack of empirical frameworks for the automated, quantitative evaluation of TPL behaviour in mobile environments.

2.4 SUMMARY

This chapter depicted the state of the art for our three key questions, namely:

(1) Storage and protection of smartphone-produced data;

(2) Direct, smartphone-to-smartphone communication;

(3) Impact of software choices on smartphones.

DATA PROTECTION AND STORAGE. Location data is sensitive and, as such, can be protected by Location Privacy Protection Mechanisms (LPPMs), ensuring the privacy/utility trade-off is met before sharing data with third parties. Such mechanisms, however, require either storing user traces or complete location datasets, which is currently not possible. Additionally, no LPPM has yet been deployed in a mobile environment.

LOCAL COMMUNICATION. Operating communication through the cloud raises privacy and energy concerns, to which the local Device-to-device (D2D) communication model is an answer that also improves communication security properties. However, communication nowadays goes through the Internet, even when not strictly needed: this contrasts with the count of local communication channels and frameworks widely deployed on smartphones.

SOFTWARE IMPACT ON ENERGY CONSUMPTION. The majority of Android apps, whether in size or in the number of method calls, are composed of Third Party Libraries (TPLs). When choosing among many TPLs while building an Android application, developers often rely on metrics such as the number of stars or Play Store downloads, underestimating the impact of their integration on network usage or battery draw.

This thesis explores each one of these research issues:

(1) Taking location data as a use case, we show in Chapter 3 that local data processing is needed for privacy reasons but also challenging, and propose tools to enable local data processing and storage;

(2) In Chapter 4, we explain why the D2D communication class is underused, presenting the results of a survey, and propose a tool to help developers adopt this paradigm;

(3) Finally, we investigate the impact of Third Party Libraries (TPLs) in Chapter 5 and propose an evaluation protocol to allow TPL comparison and thus TPL choice, limiting their impact on Android apps.

3

TOWARDS EFFICIENT ONLINE DATA STORAGE THROUGH PIECEWISE LINEAR COMPRESSION

This thesis defends the idea of storing and processing data locally to protect users' privacy. Unfortunately, this vision is hampered by the limited storage capacity of many user-side devices.

In this chapter we propose Fast Linear Interpolation (FLI), leveraging linear functions to store time series, in Section 3.1; then, we present Fast Linear Interpolation in n-Dimensions (FLInD), an extension of the previous contribution adding support for multidimensional series, in Section 3.2; finally, we integrate these storage components into the IN-siTu locAtion proteCTion (INTACT) tool, a privacy protection framework running directly on smartphones, in Section 3.3.

Contents

3.1 Compact storage of data streams **24**
3.1.1 FLI: Online Time Series Approximation . . 24
3.1.2 Experimental Setup 29
3.1.3 Experimental Results 33
3.2 A lossy storage middleware for multivariate time series **39**
3.2.1 Overview 39
3.2.2 Choice of parameters 40
3.2.3 Algorithm walk-through 42
3.2.4 Experimental setup 43
3.2.5 Results 47
3.3 Protecting user privacy through local data storage **51**
3.3.1 Divide&Stay: Attacking Location Privacy . 52
3.3.2 Enabling *in-situ* big data processing 54
3.4 Application of a predictive method to protect the privacy of mobility data **58**
3.5 Discussion **59**
3.5.1 Threats to validity 59
3.5.2 Future works 60
3.5.3 Perspectives 61
3.5.4 Conclusion 62

3.1 COMPACT STORAGE OF DATA STREAMS

To overcome the memory constraints of mobile devices, we claim that efficient temporal storage solutions must be ported onto ubiquitous environments. In particular, we advocate the use of data approximation, such as Piecewise Linear Approximation (PLA) [78, 108] or GREYCAT [142], to increase the storage capacity of mobile devices.

In the following, we present two contributions that exploit this intuition. FLI decomposes a Univariate Time Series (UTS) into a set of segments, and FLInD extends FLI's internal format to capture Multivariate Time Series (MTS).

3.1.1 *FLI: Online Time Series Approximation*

An online storage approach should meet specific requirements: fast read/write operations, low compression error, and the ability to add new data points incrementally without recompressing the whole series. Existing approaches do not conform to this criteria, notably the I/O speed concern.

We subsequently introduce FLI, an online time-series modelling algorithm based on an iterative and continuous PLA that stores approximate models of data streams in memory-constrained devices, rather than storing all raw data samples as state-of-the-art temporal databases do.

The intuition behind FLI is that most time series generally do not vary abruptly but rather follow smooth trends with quasilinear portions, whether their values increase, decrease, or remain the same over time. Since linear segments are suitable for representing linear trends, one can imagine a given time series as a sequence of segments, with each segment capturing as many original data points as possible while keeping the error below a predefined threshold. FLI does just that: given a configuration parameter $\epsilon \in \mathbb{R}^{+*}$, it captures any univariate time series as a series of segments (or interpolations), dropping original data points (or samples) as long as the following invariant is preserved:

> *All samples captured by an interpolation maintain an error below the maximum threshold ϵ.*

FLI is an online algorithm: each data point from the original time series is inserted in order. During this process, FLI decides whether the latest sample can be modelled with the latest interpolation or if a new segment needs to be created. We will first present FLI's representation of the time series, then how reads are performed, before diving into the insertion algorithm.

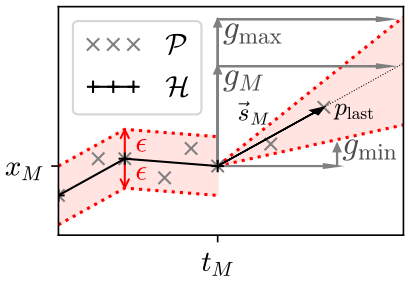


Figure 3.1: A time series $\mathcal{P}$ being modelled with FLI, displaying every component of FLI's data structure $\mathcal{F} = (\mathcal{H}, g_M, g_{\min}, g_{\max}, p_{\text{last}})$. $\mathcal{H}$ models every point in $\mathcal{P}$ while keeping their error below ϵ. $\mathcal{H}$'s last entry, $\mathcal{H}[M] = (t_M, x_M)$, is the origin of the latest model $\vec{s}_M$, with slope g_M. $\vec{s}_M$ can be used to read past $\mathcal{H}[M]$ and into the future. Gradients $g_{\min}$ and $g_{\max}$ bound the next sample insertions so as to keep their modelling error below ϵ. The three slopes are updated upon insertion. The last inserted point p_{last} is also saved for next iterations.

FLI'S MATHEMATICAL FORMULATION Consider an ever-growing univariate real-valued time series $\mathcal{P} \subset \mathbb{R}^2$, such that its i-th point is expressed as: $\mathcal{P}[i] = (t_i, x_i)$, with $i \in \mathbb{N}^*$, where $t_i \in \mathbb{R}^+$ is the point's timestamp, and $x_i \in \mathbb{R}$ is its value. Let $\mathcal{F}$ be the data structure containing FLI's representation of $\mathcal{P}$ under an error bound of ϵ. $\mathcal{F}$ is such that for each time stand t_i it is able to produce an estimation $\mathcal{F}(t_i)$ of the value x_i so that for all t_i, $|\mathcal{F}(t_i) - x_i| \leq \epsilon$. We say that $\mathcal{F}$ *models* $\mathcal{P}$ under ϵ, or: $\mathcal{F} \models_\epsilon \mathcal{P}$.

The data structure $\mathcal{F}$ is composed of *i)* a series of historical points $\mathcal{H}$ selected from $\mathcal{P}$, *ii)* a triplet of gradients $(g_M, g_{\min}, g_{\max})$ that are used for reading and insertion, and *iii)* the last inserted point p_{last}. How these elements are used is displayed in Fig. 3.1.

$$\mathcal{F} = (\mathcal{H}, g_M, g_{\min}, g_{\max}, p_{\text{last}}) \text{ s. t.}$$
$$\begin{cases} \mathcal{H} \subset \mathcal{P} \subset \mathbb{R}^2 \\ (g_M, g_{\min}, g_{\max}) \in \mathbb{R}^3 \\ p_{\text{last}} \in \mathbb{R}^2 \end{cases}$$

Any *segment* $\vec{s}$ in $\mathcal{H}$ *models* one or more points in the original time series $\mathcal{P}$:

$$\begin{cases} \forall k \in [\![1, |\mathcal{H}|[\![\\ \exists\,(i,j) \in [\![1, |\mathcal{P}|]\!]^2 \\ i < j \end{cases} \text{ s. t. } \begin{cases} \mathcal{H}[k] = \mathcal{P}[i] \\ \mathcal{H}[k+1] = \mathcal{P}[j] \\ \vec{s}_k = \langle \mathcal{H}[k], \mathcal{H}[k+1] \rangle \end{cases} \text{ then}$$

$$\forall l \in [\![i, j[\![\,, \vec{s}_k \models_\epsilon \mathcal{P}[l] \iff \text{dist}(\vec{s}_k, \mathcal{P}[l]) < \epsilon \tag{3.1}$$

with *dist* individually comparing $\vec{s}_k$ and $\mathcal{P}[l]$ members.

The latest (or *current*) segment $\vec{s}_M$ is treated differently: as it starts from the last point in $\mathcal{H}$, FLI stores its slope g_M in $\mathcal{F}$:

Algorithm 1 Approximate read using FLI

Require: $\begin{cases} \mathcal{F} = (\mathcal{H}, g_M, g_{\min}, g_{\max}, p_{\text{last}}) \\ M = |\mathcal{H}| \\ (t_M, x_M) = \mathcal{H}[M] \end{cases}$

1: **function** READ($t \in \mathbb{R}$)
2: **if** $t \leq t_M$ **then** ▷ Historical read
3: **find** $k \in [\![1, M[\![$ **s. t.** $\begin{cases} \mathcal{H}[k].t \geq t \\ \mathcal{H}[k+1].t < t \end{cases}$
4: $g \leftarrow \frac{\mathcal{H}[k+1].x - \mathcal{H}[k].x}{\mathcal{H}[k+1].t - \mathcal{H}[k].t}$ ▷ Compute img. on $\vec{s}_k$
5: **return** $g \times (t - \mathcal{H}[k].t) + \mathcal{H}[k].x$
6: **else** ▷ Forward read
7: **return** $g_M \times (t - t_M) + x_M$
8: **end if**
9: **end function**

$$\begin{cases} M = |\mathcal{H}| \\ \exists i \in [\![1, |\mathcal{P}|]\!] \end{cases} \text{s. t.} \begin{cases} \mathcal{H}[M] = \mathcal{P}[i] \\ \vec{s}_M = \langle \mathcal{H}[M], g_M \rangle \end{cases} \text{then}$$

$$\forall l \in [\![i, |\mathcal{P}|]\!], \vec{s}_M \models_\epsilon \mathcal{P}[l] \iff \text{dist}(\vec{s}_M, \mathcal{P}[l]) < \epsilon \tag{3.2}$$

READING DATA STREAMS In FLI, reading a value at time t is achieved by estimating its image using the appropriate interpolation, as detailed in Algorithm 1.

If t is lower than the last timestamp stored in the history (t_M for short), then a position on an interpolation stored in $\mathcal{H}$ will be retrieved. Line 3 selects the appropriate index k such that $\vec{s}_k = \langle \mathcal{H}[k], \mathcal{H}[k+1] \rangle$ contains the queried timestamp: $\mathcal{H}[k].t \leq t < \mathcal{H}[k+1].t$. In practice, the lookup is done with a dichotomy search. Lines 4-5 return the image of t on the segment $\vec{s}_k$, after having computed the gradient g between the consecutive historical points $\mathcal{H}[k]$ and $\mathcal{H}[k+1]$.

If t is ulterior or equal to t_M, line 7 returns the image on the current interpolation $\vec{s}_M = \langle \mathcal{H}[M], g_M \rangle$.

INSERTING DATA SAMPLES Let us now go through the most important algorithm in FLI, the sequential insertion, which is in charge of maintaining the invariant: *all modelled points in $\mathcal{P}$ have an approximation error below ϵ.* Data samples are inserted sequentially: the current interpolation is adjusted to fit the new samples until it no longer satisfies the invariant. A naive solution to maintain the invariant while updating the current model would be to memorise every sample between $\mathcal{H}[M]$ and the last observed sample, to check their error against the model—which would be costly. Instead, FLI only maintains $g_{\min}$ and $g_{\max}$, which are cost-effectively updated at each sample insertion.

Algorithm 2 Insertion using $\epsilon \in \mathbb{R}^{+*}$

Require: $\begin{cases} \mathcal{F} = (\mathcal{H}, g_M, g_{\min}, g_{\max}, p_{\text{last}} = (t_{\text{last}}, x_{\text{last}})) \\ M = |\mathcal{H}| \\ (t_M, x_M) = \mathcal{H}[M] \end{cases}$

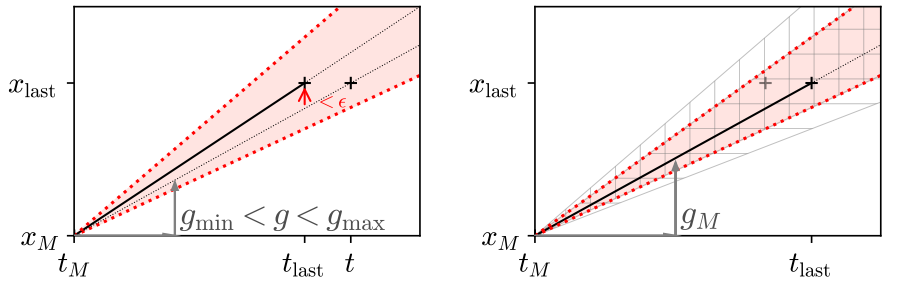
1: **function** INSERT($p = (t, x) \in \mathbb{R}^2$, ϵ)
2: $(t_\Delta, x_\Delta) \leftarrow (t - t_M, x - x_M)$
3: $g \leftarrow x_\Delta / t_\Delta$ ▷ Compute g
4: **if** $g_{\min} < g < g_{\max}$ **then**
5: $g_M \leftarrow g$ ▷ Update model (Fig. 3.2)
6: $\left(g^p_{\min}, g^p_{\max}\right) \leftarrow \left(\frac{x_\Delta - \epsilon}{t_\Delta}, \frac{x_\Delta + \epsilon}{t_\Delta}\right)$
7: $g_{\min} \leftarrow \max\left(g_{\min}, g^p_{\min}\right)$
8: $g_{\max} \leftarrow \min\left(g_{\max}, g^p_{\max}\right)$
9: **else**
10: $\mathcal{H} \leftarrow \mathcal{H} \cup [(t_{\text{last}}, x_{\text{last}})]$ ▷ New model (Fig. 3.3)
11: $(t'_\Delta, x'_\Delta) \leftarrow (t - t_{\text{last}}, x - x_{\text{last}})$
12: $g_M \leftarrow x'_\Delta / t'_\Delta$
13: $g_{\min} \leftarrow (x'_\Delta - \epsilon) / t'_\Delta$
14: $g_{\max} \leftarrow (x'_\Delta + \epsilon) / t'_\Delta$
15: **end if**
16: $(t_{\text{last}}, x_{\text{last}}) \leftarrow (t, x)$ ▷ Update last sample
17: **end function**

Notation-wise, we will consider than the original series $\mathcal{P}$ is ever-growing, and that at each insertion, FLI handles sample $\mathcal{P}[|\mathcal{P}|]$, noted $p = (t, x)$. The last point inserted in FLI's data structure $\mathcal{F}$ is noted $p_{\text{last}} = (t_{\text{last}}, x_{\text{last}})$. Before insertion, p and p_{last} coexist. After insertion, p_{last} takes the value of p.

Following along Algorithm 2, we will first cover the insertion of samples that fall within the current interpolation (represented in Fig. 3.2), before explaining how a new segment is generated once the last point breaks the invariant (displayed in Fig. 3.3).

Upon insertion of a new sample p, lines 2-3 first compute the slope g of the segment $\vec{s} = \langle \mathcal{H}[M], p \rangle$. On line 4, g is compared to the interval $]g_{\min}, g_{\max}[$, to know whether p falls within the bounds of the current interpolation $\vec{s}_M$ or not.

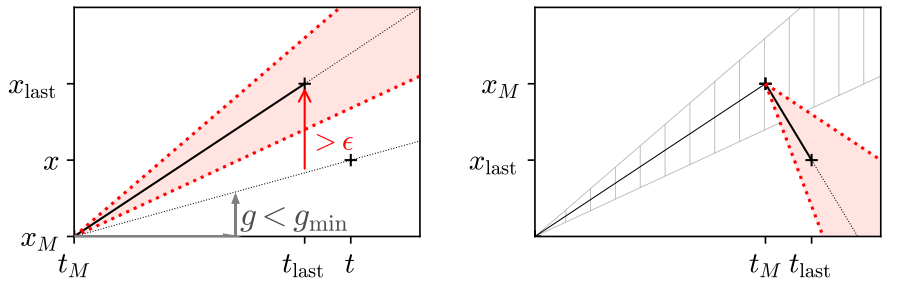
If it falls within (cf. Fig. 3.2), p is added to the current interpolation: g_M is updated to g on line 5, g_M being used for reading ahead (see Alg. 1). Two new slopes $g^p_{\min}$ and $g^p_{\max}$ are computed on line 6. They respectively represent the lines $\langle \mathcal{H}[M], (t, x - \epsilon) \rangle$ and $\langle \mathcal{H}[M], (t, x + \epsilon) \rangle$. Together, those two lines materialise p's *allowed cone* for future values, so as to keep $\mathcal{H}[M]$ as the origin of the model while preserving the invariant for p and the future insert. On line 7, $g_{\min}$ takes the maximum between its previous value and $g^p_{\min}$; on line 8, the converse operation happens to $g_{\max}$. What these two lines



(a) p's gradient g remains within $[g_{\min}, g_{\max}]$.

(b) The model is updated to encompass p.

Figure 3.2: When a new sample fits within $[g_{\min}, g_{\max}]$, it is added to the current model by updating $g_{\min}$ and $g_{\max}$. g_M is also updated for read queries (see Alg. 1).



(a) The error of p exceeds ϵ.

(b) A new model is created from the last two points.

Figure 3.3: When an error $> \epsilon$ is reported, a new model is created using p_{last} as its origin.

do is *shrinking* the interval $]g_{\min}, g_{\max}[$ so as to encompass p's allowed cone in the next iteration. Recursively, $g_{\min}$ and $g_{\max}$ represent the current model's allowed cone, which is *the intersection of all modelled points' allowed cones*—similarly to [59], but in an online fashion. The process is pictured in Fig. 3.2b: vertical hatches represent the previous point's allowed cone, horizontal ones show the last point's, while the model's allowed cone is represented in red.

If g falls outside $]g_{\min}, g_{\max}[$ (that is: if p falls outside the model's allowed cone), then a new interpolation must begin from the two last points (see Fig. 3.3). It begins from the last two observed samples: the new $\vec{s}_M$ becomes $\langle p_{\text{last}}, p \rangle$. The penultimate sample p_{last} is thus persisted to the series of selected points $\mathcal{H}$ at line 10, and new values for $g_M, g_{\min}$ and $g_{\max}$ are derived based on the new $\vec{s}_M$ (lines 11–14). Recognise here how the computation of the new gradients is equivalent to the computation of p's allowed cone on line 6 (with a different origin).

In any case, the penultimate sample p_{last} is updated on line 16.

THE CASE OF ϵ. The value of the error bound parameter significantly affects FLI's performance. If ϵ is too small, none of the inserted

samples fit the current model at that time, thus initiating a new linear fragment each time. In that case, there will be one segment per sample, imposing an important memory overhead. The resulting model overfits the data. On the other hand, if ϵ is too large, then all the inserted samples fit, and a single segment is kept. While it is the best case memory-wise, the resulting linear fragment simply connects the first and last points and underfits the data.

The choice of ϵ is highly dependent on the input data. For instance, accelerometer data—which generally has a high sampling rate and little variance—is captured efficiently even with small values of ϵ. Whereas GPS data — low sampling rate and high variance — require finer tuning. We propose a parameter tuning process in Section 3.1.2.

On the other hand, the cautious reader will have observed how ϵ is only a parameter of the INSERT function in Alg. 2. In future works, our team intends to dynamically adapt this error bound depending on the data and its freshness.

3.1.2 *Experimental Setup*

Key Performance Metrics.

Memory footprint. The key objective of FLI is to reduce the memory footprint required to store an unbounded stream of samples. We explore two metrics: *(i)* the number of 64-bit variables required by the model and *(ii)* the size of the model in the device memory. To do so, we compare the size of the persistent file with the size of the vanilla SQLITE database file. We consider the number of 64-bit variables to be a device-agnostic estimate of the model footprint.

I/O throughput. Another key system metric is the I/O throughput of the temporal databases. In particular, we measure the number of write and read operations per second (IOPS).

Input Datasets & Parameter Tuning.

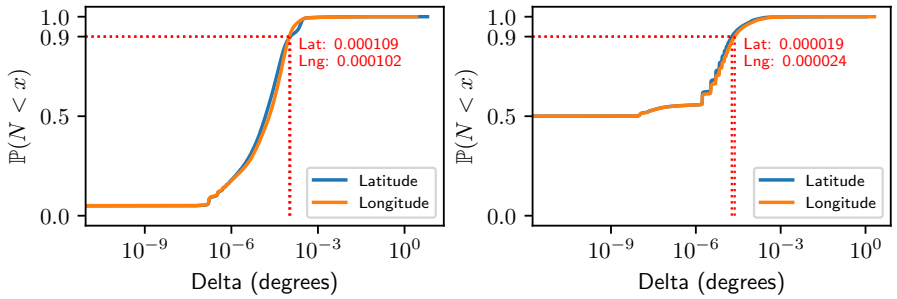
FLI was built to allow the storage of user-generated data series, such as GPS and accelerometer streams (although it is readily applicable to other types of real-valued streams, as shown in Section 3.1.3). However, the choice of the ϵ parameter depends on the underlying data distribution. Towards that end, we propose a semi-automated routine for selecting the ϵ parameter.

In this Section, we present the two datasets that will be used throughout our evaluation, before outlining and applying our parameter selection algorithm to the both of them.

LOCATION DATASETS. We use two real-world mobility datasets that exhibit different characteristics:

- CABSPOTTING [163] is a mobility dataset of 536 taxis in the San

[163] Piorkowski et al. (2022), "CRAWDAD EPFL/MOBILITY"



(a) CDF of CABSPOTTING locations' deltas. (b) CDF of PRIVAMOV locations' deltas.

Figure 3.4: CDF of latitude and longitude deltas (in degrees) of successive location samples in CABSPOTTING and PRIVAMOV datasets. One can observe that, from one location sample to the next, latitude or longitude deviations are small.

Francisco Bay Area. The data was collected over a month and comprises 11 million records, totalling 388 MB. It consists solely of car trips in a dense urban environment.

- PRIVAMOV [144] is a multi-sensor mobility dataset gathered during 15 months by 100 users around the city of Lyon, France. It contains several transportation modes, most notably pedestrian. PRIVAMOV displays a much higher sampling rate that CABSPOTTING. We use the full GPS dataset, which includes 156 million records and totals 7.2 GB.

[144] Mokhtar et al. (2017), "PRIVA'MOV: Analysing Human Mobility Through Multi-Sensor Datasets"

PARAMETER TUNING. The choice of an ϵ value is of major importance and plays a central role in FLI's performance: A poorly chosen value has a significant impact on FLI's underlying segments, either degrading modelled data quality or filling storage space unnecessarily. To find a compromise between the two, since the ϵ value is highly correlated with the modelled data, one has to *know* the data; more specifically, we propose studying the signal's amplitude variation between consecutive values.

For example, in the context of location data, Fig. 3.4 characterises —as a CDF—the evolution of longitude and latitude samples for all the traces stored in the CABSPOTTING and PRIVAMOV datasets. In particular, we plot the CDF of the drift d observed between 2 consecutive values (t_1, x_1) and (t_2, x_2), which we compute as $d = |x_2 - x_1|/|t_2 - t_1|$. One can observe that CABSPOTTING and PRIVAMOV datasets report on a drift lower than 1×10^{-4} and 2×10^{-5} for 90% of the values, respectively. Furthermore, due to the high density of locations captured by PRIVAMOV, half of the drifts are equal to 0, meaning several consecutive longitudes or latitudes are unchanged. This preliminary analysis shows that FLI can efficiently model mobility data and demonstrates that $\epsilon = 10^{-3}$ is a conservative choice for these datasets.

To automate the tuning of ϵ, one must input a sample of their data stream into our provided script[1], which proposes candidate ϵ parameter values to capture 90%, 95% and 99% of the sampled data.

Baselines.

We compare FLI to three existing approaches: the SQLite relational database, and the SWAB and Greycat regression algorithms.

SQLite is the state-of-the-art solution to persist and query large volumes of data on Android devices. SQLite provides a lightweight relational database management system. SQLite is not a temporal database, but is a convenient and standard way to store samples persistently on a mobile device. Insertions are atomic, so one may batch them to avoid one memory access per insertion.

Sliding-Window And Bottom-up (SWAB) [108] is a linear interpolation model. As FLI, the samples are represented by a list of linear models. In particular, reading a sample is achieved by iteratively traversing the list of models until the corresponding one is found, then using it to estimate the requested value. The bottom-up approach of SWAB starts by connecting every pair of consecutive samples, then iterates by merging the least significant pair of consecutive interpolations. This process is repeated until no more pairs can be merged without introducing an error higher than ϵ. Contrary to FLI, this bottom-up approach is offline, requiring all samples to be known. SWAB extends the bottom-up approach by buffering samples in a sliding window. New samples are inserted in the sliding window and then modelled using a bottom-up approach: whenever the window is full, the oldest model is kept and the captured samples are removed from the buffer.

One could expect that the bottom-up approach delivers more accurate models than the greedy FLI, even resulting in a slight reduction in the number of models and faster readings. On the other hand, sample insertion is more expensive than FLI because it requires the bottom-up approach for storing samples. Like FLI, SWAB ensures that reading stored samples is at most ϵ away from the exact values.

Greycat [142] aims at compressing even further the data by not limiting itself to linear segments. Greycat also represents the samples as a list of polynomials, but these polynomials are not necessarily linear. The samples are read the same way.

When inserting a sample, it first checks if it fits the model. If so, then nothing needs to be done. Otherwise, unlike FLI and SWAB, which directly initiate a new model, Greycat increases the degree of the polynomial to better fit the new sample. To do so, Greycat first regenerates $d+1$ samples in the interval covered by the current model, where d is the degree of the current polynomial. Then, a polynomial

1 See file `lib/epsilon_choice.dart` in our code repository [172].

regression of degree $d+1$ is computed on those points along the new one. If the resulting regression reports an error below $\frac{\epsilon}{2^{d+1}}$, the polynomial is kept; otherwise, the process is repeated by incrementing the degree until either a fitting polynomial is found or the maximum degree is reached. If the maximum degree is reached, the former polynomial is stored, and a new polynomial is initiated. The resulting model is quite compact and thus faster to read, but at the expense of an important insertion cost.

Unlike FLI and SWAB, there can be errors higher than ϵ for the inserted samples, as the errors are not computed on raw samples but on generated ones, which may not coincide. Furthermore, the use of higher-degree polynomials makes the implementation prone to overflow; to alleviate this, the inserted values are normalised.

Experimental Settings.

For experiments with univariate data streams—*i.e.* memory and throughput benchmarks—we set $\epsilon = 10^{-2}$. The random samples used in those experiments follow a uniform distribution in $[-1{,}000; 1{,}000]$: it is very unlikely to have two successive samples with a difference lower than ϵ, hence reflecting the worst case conditions for FLI. For experiments on location data, and unless said otherwise, we set $\epsilon = 10^{-3}$ for FLI, SWAB and Greycat. For Greycat, the maximum degree for the polynomials is set to 14. The experiments evaluating the throughput were repeated 4 times each, and the average is taken as the standard deviation was low. All the other experiments are deterministic and performed once.

Implementation Details.

[69] Google (2018), Flutter framework

We implemented FLI using the Flutter *Software Development Kit* (SDK) [69]. Flutter is Google's UI toolkit, based on the Dart programming language, that can be used to develop natively compiled apps for Android, iOS, the web, and desktop platforms (as long as the project's dependencies support cross-compilation to all considered platforms). Our implementation contains 23 classes and 7,000 lines of code, including FLI and its storage competitors; it is publicly available [172].

[172] Raes et al. (2022), FLI implementation

For our experiments, we also implemented several mobile applications based on this library. To demonstrate its capability to operate across multiple environments (models, operating systems, processors, memory capacities, storage capacities), all our benchmark applications were successfully installed and executed on the devices listed in Table 3.1. Unless mentioned otherwise, the host device for the experiments is the Fairphone 3.

Table 3.1: Mobile devices used in the experiments.

Model	OS	CPU	Cores	RAM	Storage
Lenovo Moto Z	Android 8	Snapdragon 820	4	4 GB	32 GB
Fairphone 3	Android 11	Snapdragon 632	8	4 GB	64 GB
Pixel 7 Pro	Android 13	Google Tensor G2	8	12 GB	128 GB
iPhone 12	iOS 15.1.1	A14 Bionic	6	4 GB	64 GB
iPhone 14 Plus	iOS 16.0.1	A15 Bionic	6	6 GB	128 GB

3.1.3 *Experimental Results*

In this Section, we evaluate our implementation of FLI on Android and iOS to demonstrate how it enables efficient data-stream storage on mobile devices. We first perform several benchmarks (memory, throughput, & stability), before evaluating the performance of FLI beyond location streams.

Memory Benchmark.

As there is no temporal database (e.g., InfluxDB) available on Android, we compare FLI's performances with SQLite, the only database natively available on Android.

For this comparison, we use several data sources: synthetic data, and two sets of mobility traces that are representative of the type of data FLI might be applied to: the Cabspotting [163] dataset, and the PrivaMov [144] dataset.

Synthetic data. Two identical operations are performed with SQLite and FLI: *(i)* the incremental insertion of random samples and *(ii)* the incremental insertion of constant samples. The memory footprint of both solutions on disk is compared when storing timestamped values. As FLI models the inserted samples, random values represent the worst-case scenario it can face, while inserting constant values represent the ideal case. One million samples are stored and, for every 10,000 insertion, the file size associated with the storage solution is reduced. The experiments are done with a publicly available application [174].

[174] Raes et al. (2022), Memory space benchmarking application

Fig. 3.5 depicts the memory footprint of both approaches. On the one hand, the size of the SQLite file grows linearly with the number of inserted samples, no matter the nature (random or constant) of the samples. On the other hand, the FLI size grows linearly with random values, while the size is constant for constant values. In particular, for the constant values, the required size is negligible. The difference between vanilla SQLite and FLI is explained by the way the model is stored: while SQLite optimises the way the raw data is stored, FLI is an in-memory stream storage solution, which naively stores coefficients in a text file. Using more efficient storage would further shrink the difference between the two. As expected, the memory

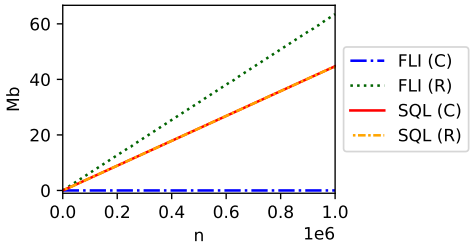


Figure 3.5: Compressed storage cost after inserting $1M$ samples, random (R) or constant (C), in SQLite and FLI.

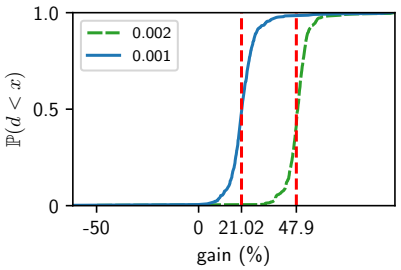


Figure 3.6: Memory gain distribution when storing Cabspotting with FLI.

footprint of a data stream storage solution outperforms that of a vanilla SQLite database in the case of stable values. While random and constant values are extreme cases, in practice, data streams produced by ubiquitous devices exhibit behaviour between the two scenarios, allowing FLI to reduce the memory required to store them.

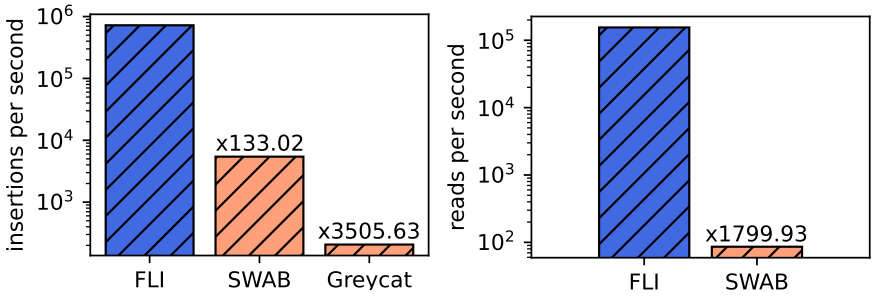
GPS data. We use FLI to store latitudes and longitudes of the entire Cabspotting dataset (388MB) in memory, using both $\epsilon = 10^{-3}$ and $\epsilon = 2 \times 10^{-3}$ (representing an accuracy of approximately a hundred meters). For each user, we compute the memory storage gain as a percentage compared to storing the raw traces. Fig. 3.6 reports on the gain distribution as a CDF along with the average gain on the entire dataset. Most of the user traces largely benefit from using FLI, and FLI provides an overall gain of 21% (307MB) for $\epsilon = 10^{-3}$ on the entire dataset, and a gain of 47.9% (202MB) for $\epsilon = 2 \times 10^{-3}$.

Additionally, we compare SQLite and FLI for storing the entire PrivaMov dataset (7.2GB). In this context, FLI only requires 25MB (a gain of 99.65%) compared to more than 5GB (a gain of 30.56%) for SQLite, despite the naive storage scheme used by FLI. Furthermore, with smartphones featuring limited RAM (cf. Table 3.1) and not allocating the whole of it to a single application, FLI enables loading complete datasets in memory to be processed: on mobile devices, loading the raw PrivaMov dataset in memory crashes the application (due to out-of-memory errors), while FLI succeeds in fitting the full dataset into RAM. This capability is particularly interesting for enabling the deployment of data stream processing tasks on mobile devices without incurring any processing overhead.

Throughput Benchmark.

Since SQLite is unable to fit the datasets, we exclude it from the list of competitors: we now compare FLI with its competitors in the temporal database space, SWAB and Greycat.

We study the throughput of each approach in terms of IOPS. Insertion speed is computed by inserting $1M$ random samples (that is, each solution's worst-case scenario). For the reads, we also incrementally



(a) Sequential Insertions (b) Random Reads

Figure 3.7: Throughput for insertions and reads using FLI, SWAB, and GREYCAT (log scale). FLI drastically outperforms its competitors for insertions and reads.

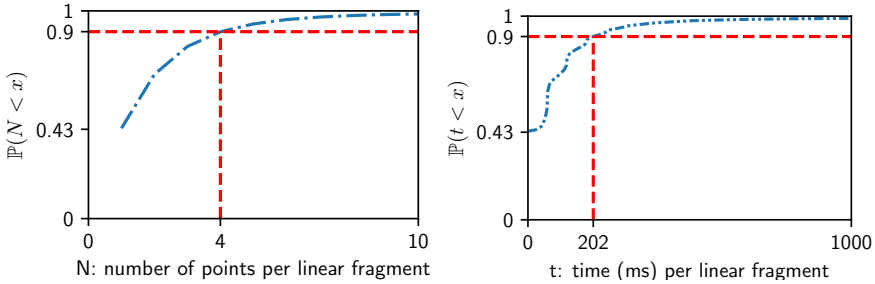
insert $1M$ samples before querying $10K$ random samples from them. GREYCAT is an exception: due to its long insertion time (Sect. 3.1.2), we only insert $10K$ random values and those values are then queried. Our experiment is done using a publicly available application [175].

[175] Raes et al. (2022), Throughput benchmarking application

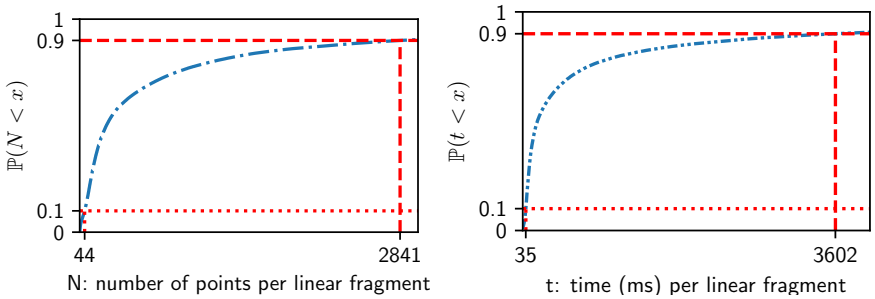
Fig. 3.7 depicts the throughput of the approaches for sequential insertions and random reads. On the one hand, FLI drastically outperforms its competitors for the insertions: it provides a speed-up from $\times 133$ against SWAB up to $\times 3{,}505$ against GREYCAT. The insertion scheme of FLI is as fast as it relies on a few parameters. On the other hand, GREYCAT relies on a costly procedure when a sample is inserted: it increases the degree of the current model until it fits the new point or reaches a maximum degree. GREYCAT aims at computing a model as compact as possible, which is not the best choice for fast online insertions.

For the reads (Fig. 3.7b), FLI also outperforms SWAB. Our investigation reports that FLI largely benefits from its dichotomy lookup inside the time index (see Alg. 1), compared to SWAB, which scans the list of models sequentially until the correct time index is found. SWAB reads have a complexity that is linear in the list size, while FLI has a logarithmic complexity. GREYCAT follows the same approach as SWAB, which is why it is not included in the results: with only $10K$ insertions instead of $1M$, its list of polynomials is significantly smaller than the others', making the comparison unfair. Nevertheless, we expect GREYCAT to achieve better throughput, as its polynomials list will be shorter.

Note that those results have been obtained with the worst-case: random samples. Similarly, FLI is unfit for periodical signals, such as raw audio: our tests show memory usage similar to that of random noise. Because FLI leverages linear interpolation, it performs best with signals that have a linear shape (e.g., GPS and accelerometer signals). We expect SWAB to store fewer segments than FLI thanks to its sliding window, resulting in faster reads. However, the throughput obtained for FLI is minimal and FLI is an order of magnitude faster than SWAB



(a) Samples per CABSPOTTING fragment. (b) Duration of CABSPOTTING fragments.



(c) Samples per PRIVAMOV fragment. (d) Duration of PRIVAMOV fragments.

Figure 3.8: Cumulative distribution functions depicting the stability of the FLI linear fragments on PRIVAMOV & CABSPOTTING with $\epsilon = 10^{-3}$.

for insertions, so it does not make a significant difference. We can conclude that FLI is the best solution for storing large streams of data samples on mobile devices.

Stability Benchmark.

We further explore FLI 's ability to capture stable linear fragments that group as many data samples as possible for the longest possible durations. Fig. 3.8 reports on the time and the number of samples covered by the linear fragments of FLI for the CABSPOTTING and PRIVAMOV datasets. One can observe that the stability of FLI depends on the density of the considered datasets. While FLI only captures at most 4 samples per segment for 90% of the linear fragments stored in CABSPOTTING (Fig. 3.8a), it reaches up to 2,841 samples in the context of PRIVAMOV (Fig. 3.8c), which samples GPS locations at a higher frequency than CABSPOTTING. This is confirmed by Fig. 3.8b and 3.8d, which report a time coverage of 202 ms and 3,602 ms for 90% of FLI linear fragments in CABSPOTTING and PRIVAMOV, respectively. Given that PRIVAMOV is a larger dataset than CABSPOTTING (7.2 GB vs. 388 MB), one can conclude that FLI succeeds in scaling with the volume of data to be stored.

Beyond Location Streams.

In this paper, FLI was primarily tested on location streams, but our proposal efficiently approximates signals that vary mostly linearly: times-

tamps, accelerations, temperature, pressure, humidity, light, proximity, air quality, etc. This makes FLI a valuable candidate for ubiquitous contexts, as many physical quantities captured by, e.g., IoT sensors exhibit piecewise linear behaviour. To showcase different scenarios, we benchmark the storage of timestamps, device accelerations and heartbeat data using FLI. In any case, we use our ϵ-tuning script to determine the most appropriate value for storing the data with reduced error.

STORING TIMESTAMPS. In all previous experiments, FLI did not store timestamps, as we expect the user to query the time at which she is interested in the samples. However, it is straightforward to store irregular timestamps using FLI: we store pairs (i, t_i), where t_i is the i^{th} timestamp inserted. The nature of the timestamps makes them a good candidate for FLI storage, as insertion rates are generally fixed or vary linearly.

To assess the efficiency of FLI for storing timestamps, we stored all timestamps for `user 1` in the PRIVAMOV dataset with $\epsilon = 1$—*i.e.*, we tolerate an error of 1 second per estimate. The 4,341,716 timestamps were stored using 26,862 linear segments for a total of 80,592 floats and an overall gain of 98%, with a *mean average error* (MAE) of 0.246 second. Hence, not only does the use of FLI result in drastic memory savings, but it also provides accurate estimations.

STORING DEVICE ACCELERATIONS. Accelerometer data is important for many context-aware applications, including transport mode detection [55, 217, 232]. Coupled with GPS information, it is possible to infer whether the user is walking, biking, driving, or taking a tram. However, storing the output of a mobile accelerometer is particularly challenging, as it generates hundreds of noisy 3D samples per second. Our implementation is publicly available [176].

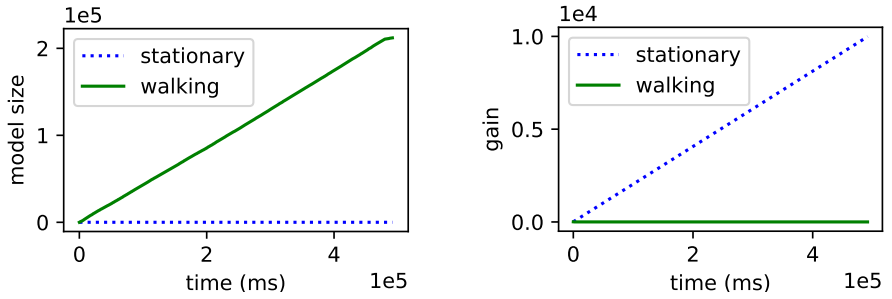
We store 10,000 consecutive accelerometer samples with FLI and, for every 100 insertions, report the file size and the relative gain. We use FLI with $\epsilon = 1$ as the accelerometer has high variability, even when the mobile is stationary. Fig. 3.9 reports on a constant memory footprint of FLI during the experiment, while providing a high-level accuracy.

[217] Wang et al. (2019), "Enabling reproducible research in sensor-based transportation mode recognition with the Sussex-Huawei dataset"

[176] Raes et al. (2022), Fast Linear Interpolation (FLI) accelerometer example application

STORING HEARTBEAT PULSES. We downloaded pulse-to-pulse intervals, which oscillate between 500 and 1,100 ms and are reported as the time duration between cardiac pulses to the timestamp of the original pulse, from a *Polar Ignite 2* [165] smartwatch, gathering 28,294,762 samples covering the 12 months of 2023, as a file of 259.1 MB. Using FLI to compress this dataset with an accuracy of $\epsilon = 100$ reports on a non-negligible storage space gain of 26.44%, with a *mean error* of 22.74 milliseconds.

[165] Polar (2021), Ignite 2



(a) Model size (b) Memory gain

Figure 3.9: Model size and gain of FLI storing accelerometer values ($\epsilon = 1$). FLI reports a constant memory whenever stationary, and a small gain ($> \times 1.39$) when walking.

FLI is thus a suitable solution to store data streams produced by various sensors of wearable and mobile devices, which could find application in e.g. human context recognition [212].

[212] Vaizman et al. (2017), "Recognizing Detailed Human Context in the Wild from Smartphones and Smartwatches"

• • •

We have seen in the past Section how FLI is useful to store temporal data: supporting Univariate Time Series (UTS) by default, several FLI instances can be used to capture multiple dimensions of a given series. We will now see how FLI can be improved to natively support Multivariate Time Series (MTS).

3.2 A LOSSY STORAGE MIDDLEWARE FOR MULTIVARIATE TIME SERIES

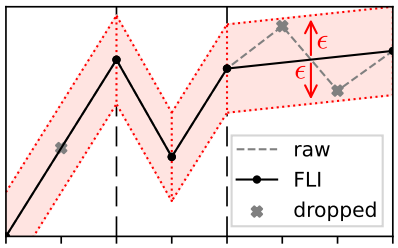


Figure 3.10: Demonstration of *univariate* FLI [180]. The original time series $\mathcal{P}$ (in grey) is compressed into its FLI representation $\mathcal{H}$ (in solid black): a series of linear interpolations joint by breakpoints (solid black dots). All dropped points (grey crosses) maintain an error with the FLI interpolation below the configuration parameter ϵ: they remain in the light-red region.

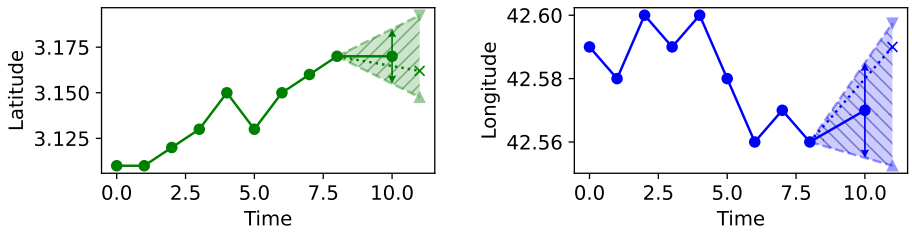
Most existing time-series compression solutions address only univariate time series. Compressing a multivariate series implies running several compression instances in parallel: one per dimension. Our intuition is that the dimensions of a multivariate time series are generally not statistically independent, and that this correlation could be leveraged to improve storage efficiency. Toward this end, we propose Fast Linear Interpolation in n-Dimensions (FLInD), a multivariate extension of the Fast Linear Interpolation (FLI) [180] (optionally lossy) compression algorithm for univariate time series, previously introduced in Section 3.1.

[180] Raes et al. (2024), "Compact Storage of Data Streams in Mobile Devices"

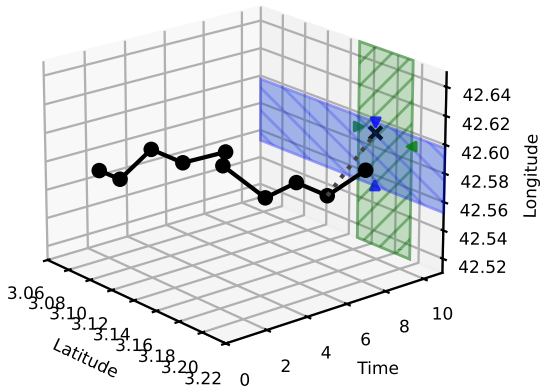
Given a real-valued error bound $\epsilon \geq 0$, FLI compresses time series in an *online* fashion by iteratively *dropping* data points in favour of a linear interpolation between their predecessor and successor, as long as the following **invariant** is preserved, which is a prolongation of FLI's original invariant: *every datum in the original series must maintain an error with its FLI interpolation that is below the error bound* ϵ. FLI's behaviour to preserve this invariant has been presented in Subsection 3.1.1; as a reminder, Figure 3.10 displays how FLI behaves with a non-null ϵ.

3.2.1 *Overview*

The contribution we present in this section, FLInD, augments FLI to support native compression of multivariate (or *nD*) time series. In essence, our aim is to leverage the fact that the random variables captured in each dimension of an nD time series tend to be statistically *dependent*. As pictured in Fig. 3.11, the goal of FLInD is to generate an nD interpolation of the whole time series (Fig. 3.11b), instead of computing one interpolation per dimension (Fig. 3.11a).



(a) Separate compression of two dimensions.



(b) Combined compression of two dimensions.

Figure 3.11: Projection of the thresholds for uni-dimensional and multi-dimensional instances, fit with latitude and longitude toy data.

MATHEMATICAL NOTATIONS A multivariate time series $\mathcal{S}$ of dimension n is represented by a couple $\mathcal{S} = (\mathcal{T}, \mathcal{P})$ where $\mathcal{T} \subset \mathbb{R}$ contains the series' timestamps, and $\mathcal{P} = (P_i)_{i \in [\![1,n]\!]} \subset \mathbb{R}^n$ is the set of data points, each P_i representing a single dimension of the time series. FLInD being an *online* compression algorithm (it compresses upon data insertion), we consider $\mathcal{T}$ and P_i as ever-growing. Because each dimension P_i can represent different physical quantities with different value ranges, FLI's ϵ error bound becomes a n-sized array in FLInD: $\vec{E} = (\epsilon_i \in \mathbb{R}^+)_{i \in [\![1,n]\!]}$.

Dimensions are not necessarily all compressed together, but rather organised in *compression groups* maintained in another n-sized array $G = (\gamma_i \in \mathbb{N})_{i \in [\![1,n]\!]}$, that maps which dimension P_i belongs to which compression group.

3.2.2 *Choice of parameters*

As shown in Fig. 3.12, FLInD makes use of two main parameters: the compression groups' mapping G, and the per-dimension error bound $\vec{E}$. These parameters are data-dependent: one cannot know which of the series' dimensions are fit to be compressed together before a sample is tested; similarly, each $\epsilon_i \in \vec{E}$ requires tuning to ensure only disposable data points are effectively forgotten (as shown in Fig. 3.10).

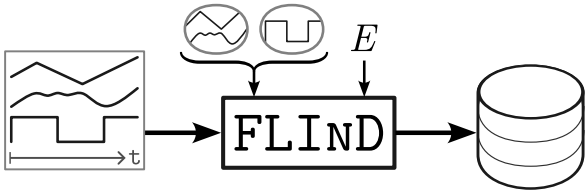
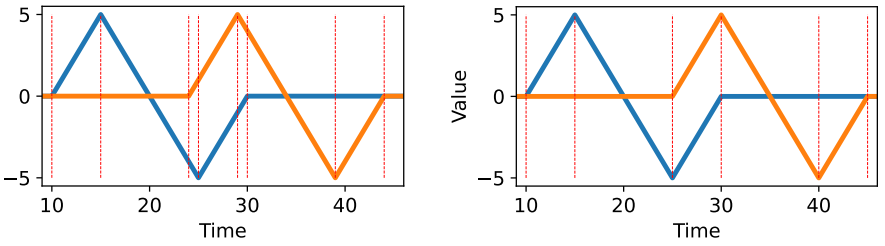


Figure 3.12: To compress a multivariate time series, FLInD requires two main parameters: the mapping of the input dimensions into different compression groups G, and the array of per-dimension error bound $\vec{E}$.



(a) Suboptimal situation for FLInD: the two series change momentum in asynchrony.

(b) Optimal situation for FLInD: both series change tendencies at the same times.

Figure 3.13: Toy example showcasing the influence of a series' alignment on FLInD's compression. Vertical, red-dotted lines mark the breakpoints where an interpolation will be recomputed. FLInD works best when breakpoints occur simultaneously across dimensions.

A parameter tuning step where a first *data sample* $\mathcal{S}' = (\mathcal{T}', \mathcal{P}')$ is analysed is thus necessary to set G and $\vec{E}$.

SETTING THE ERROR BOUND ARRAY $\vec{E}$.

First of all, let us recall that if one wants to configure FLInD for *lossless* compression, they would set ϵ_i to 0 on all dimensions and proceed. For lossy compression, though, they would need to find the right balance between compressing too little (ϵ_i too small) and compressing the signal too much (ϵ_i too large). The FLI Section dedicated paragraph 3.1.2 to the 'parameter tuning' question. In the FLInD case, we introduce parameter $\sigma \in \mathbb{R}^+$, to set the error bound to a multiple of the sample's standard deviation: $\forall i \in [\![1, n]\!], \epsilon_i = \sigma \times \text{std dev}(P'_i)$.

SETTING THE COMPRESSION GROUPS G.

None of a time series' dimensions compresses well together: it all boils down to when each dimension changes momentum, triggering a breakpoint and a new interpolation. Fig. 3.10 reads how a univariate FLI interpolation needs to be recomputed every time a steep variation occurs in the signal. In the multivariate FLInD context, when two dimensions P_i and P_j are captured together, overshooting the error bound on any dimension (ϵ_i or ϵ_j) triggers a new interpolation for both of them. In other words, dimensions of a series are optimally com-

pressible together when they tend to exhibit breakpoints at the same timestamps, as shown in Fig. 3.13: When breakpoints are misaligned, like in Fig. 3.13a, FLInD will generate more interpolations than when they are in sync, like in Fig. 3.13b. Additionally, the more dimensions in a compression group, the more often a new interpolation will be triggered, thereby reducing the global output interpolation error.

We propose several strategies to pick compression groups:

- FLInD: in its default configuration, all dimensions are compressed together: $\forall i \in [\![1, n]\!], \gamma_i = 1$. This strategy maximises the number of interpolations, thereby minimising the global interpolation error while maximising the storage footprint.
- SplitFLInD(GS): columns are sequentially put in compression groups of constant *group size*: $GS < n \mid \gamma_i = i//GS$ (i.e. γ_i takes the result of the integer division of i by GS). This simple data-independent strategy provides a simple parameter to navigate the size/precision trade-off.
- CliqueFLInD: Through clustering on the columns of the sample data $\mathcal{S}'$, the biggest compatible cluster of dimensions is compressed together, while the other dimensions are compressed individually in 1D. Section 3.2.4 presents a preliminary experiment toward this goal.

3.2.3 *Algorithm walk-through*

FLInD's online insertion algorithm, Write $(t, \vec{x})$, is displayed in Algorithm 3. It removes the notion of compression groups (G): all dimensions are compressed within a single nD instance.

As a multidimensional extension of FLI, this algorithm actually is a generalisation of FLI's insert logic (depicted by Algorithm 2 in this manuscript), where each value variable becomes a vector holding n values (*e.g.*, $g_{\min}$ now is $\overrightarrow{g_{\min}}$).

FLInD only stores data points that do not respect the bounds $\vec{E}$, triggering a new interpolation to be stored in $\mathcal{H}$. To do so, FLInD maintains three n-dimensional slope vectors: the current interpolation's direction $\overrightarrow{g_M}$ (used for reading), and its allowed bounds $\overrightarrow{g_{\min}}$ and $\overrightarrow{g_{\max}}$. It also keeps the last inserted point in $(t_{\text{last}}, \overrightarrow{x_{\text{last}}})$.

In order to insert a new point $(t, \vec{x})$, the algorithm begins at lines 2-4 by computing the n-dimensional slope $\vec{g}$ between the base of the last interpolation $(t_M, \overrightarrow{x_M})$ and the new point $(t, \vec{x})$. When the slope $\vec{g}$ of the new point remains within the bounds $[\overrightarrow{g_{\min}}, \overrightarrow{g_{\max}}]$ in all dimensions (ll. 5-8), the current interpolation's slope $\overrightarrow{g_M}$ is updated, as well as the bounds $\overrightarrow{g_{\min}}$ and $\overrightarrow{g_{\max}}$. When the slope $\vec{g}$ does exceed the bounds in any dimension (ll. 9-15), the breakpoint $(t_M, \overrightarrow{x_M})$ is persisted to the list of historical interpolations $\mathcal{H}$, a new interpolation begins with a new

Algorithm 3 Insertion in FLInD (capturing n dimensions)

Require:
- Maintained slopes: $\overrightarrow{g_M}, \overrightarrow{g_{\min}}, \overrightarrow{g_{\max}}$
- Last inserted point: $(t_{\text{last}}, \overrightarrow{x_{\text{last}}})$
- Previous interpolations: $\mathcal{H} = [(t_0, \overrightarrow{v_0}), ..., (t_M, \overrightarrow{x_M})]$
- Error bound array: $\vec{E} = (\epsilon_i \in \mathbb{R}^+)$

1: **function** WRITE($t, \vec{x} = (x_1, x_2, ..., x_n)$)
   // Compute the slope $\vec{g}$ between $(t_M, \overrightarrow{x_M})$ and $(t, \vec{x})$
2: $t_\Delta \leftarrow t - t_M$
3: $\overrightarrow{x_\Delta} \leftarrow \vec{x} - \overrightarrow{x_M}$
4: $\vec{g} \leftarrow \overrightarrow{x_\Delta} / t_\Delta$
   *// If the point remains within **all** bounds, only update the slopes*
5: **if** $\overrightarrow{g_{\min}} \leq_n \vec{g} \leq_n \overrightarrow{g_{\max}}$ **then**
6: $\overrightarrow{g_M} \leftarrow \vec{g}$
7: $\overrightarrow{g_{\min}} \leftarrow \sup\left(\overrightarrow{g_{\min}}, \frac{\overrightarrow{x_\Delta} - \vec{E}}{t_\Delta}\right)$
8: $\overrightarrow{g_{\max}} \leftarrow \inf\left(\overrightarrow{g_{\max}}, \frac{\overrightarrow{x_\Delta} + \vec{E}}{t_\Delta}\right)$
   // Else, start a new interpolation at $(t_M, \overrightarrow{x_M})$
9: **else**
10: $\mathcal{H} \leftarrow \mathcal{H} \oplus [(t_M, \overrightarrow{x_M})]$
11: $(t_M, \overrightarrow{x_M}) \leftarrow (t_{\text{last}}, \overrightarrow{x_{\text{last}}})$
12: $(t'_\Delta, \overrightarrow{x_\Delta}') \leftarrow (t - t_{\text{last}}, \vec{x} - \overrightarrow{x_{\text{last}}})$
13: $\overrightarrow{g_M} \leftarrow \overrightarrow{x_\Delta}' / t'_\Delta$
14: $\overrightarrow{g_{\min}} \leftarrow \left(\overrightarrow{x_\Delta}' - \vec{E}\right) / t'_\Delta$
15: $\overrightarrow{g_{\max}} \leftarrow \left(\overrightarrow{x_\Delta}' + \vec{E}\right) / t'_\Delta$
16: **end if**
   // $(t_{last}, \overrightarrow{x_{last}})$ always represents the last inserted point
17: $(t_{\text{last}}, \overrightarrow{x_{\text{last}}}) \leftarrow (t, \vec{x})$
18: **end function**

origin $(t_M, \overrightarrow{x_M})$ and new slopes. In any case, the last point $(t_{\text{last}}, \overrightarrow{x_{\text{last}}})$ is updated on line 17.

Please note that line 9 runs as soon as at least one dimension bound is not respected: *as opposed to running FLI n times, FLInD pairs dimensions together.*

3.2.4 *Experimental setup*

Our evaluation considers four input datasets, five competitors (in addition to FLInD), and follows the same uniform protocol to compare the different approaches between themselves.

Input datasets.

(1) *Hardware performance profile* [129] includes 66 system metrics series, like CPU or memory usage, of a laptop over several days

[129] Madsen (2022), Hardware Performance Archive [2019]

of normal functioning. In this study, we select data from the largest file (2019/05/04), containing 77,070 records sampled every second.

[210] Trindade (2015), ElectricityLoadDiagrams20112014

(2) *ElectricityLoadDiagrams20112014* [210] contains electricity consumption data from 370 Portuguese households from 2011 to 2014, with a sampling frequency of 15 minutes, totalling 140,257 records.

[155] National Renewable Energy Laboratory (2006), Solar Power Data for Integration Studies

(3) *Solar Power Data for Integration Studies* [155] contains solar power production data from several stations across the USA, organised by state; in this study, we use data from the state of California. Power produced by the 405 California solar stations is reported at a 5-minute rate for the entire year of 2006 and contains 105,120 records.

[122] Luo et al. (2021), "Characterizing Microservice Dependency and Performance: Alibaba Trace Analysis"

(4) *Microservices Traces* [122] contains normalised system metrics (CPU and memory usage) of 13,058 Alibaba bare-metal servers (in this study, we use CPU series) collected in 2021; the dataset includes 1,440 samples, collected over 12 hours at a 30-second rate.

Competitors.

In this Subsection, we present several compression competitors we evaluate FLInD against. All of the following competitors have been implemented in a publicly available notebook [177].

[177] Raes (2026), FLInD notebook

(1) The Raw storage class internally uses an in-memory dictionary that associates each timestamp with an array of values; to compute the instance's final size, one float per dataset value and per timestamp is accounted for.

(2) Our FLI [180] previous contribution leverages linear regression to store UTS, namely one series per FLI instance: using it to compress MTS is straightforward using one instance per captured dimension.

[204] Timescale Inc (2018), Timescale database

(3) TimescaleDB [204] is a lossless open-source database leveraging SQL to host up to 1,600-wide MTS.

(4) SplitFLInD: Since FLInD is the opposite case of MTS-FLI, as FLInD stores all series in one instance as opposed to one FLI instance per series, we propose to explore the in-between configuration space, and propose a "split" configuration, distributing series between several FLInD instances of fixed size.

(5) CliqueFLInD automatically computes compression groups by computing distances between all series.

FLI and FLInD both accept an optional error parameter, which tends to improve the final storage budget at the cost of degraded data accuracy. Since linear regression can provide a storage cost gain by itself when compressing linearly evolving data, we benchmark two versions of both algorithms: one with no input error, and another with an *error of one standard deviation per compressed series*.

Series dimensions' clustering.

To try and be more subtle in the compression of time series, we explore CliqueFLInD, an additional competitor which compares series together, searching for clusters of series dimensions that compress well together, thus improving the final storage budget (see the choice of compression groups G in Sec. 3.2.2).

To take advantage of combined compression, we need a way to assess the *collinearity* of series, and to store them together if they *fit*, or separately if they do not. Pearson and Spearman correlation coefficients are widely used to compare series and measure linear correlation, but do not detect shifted series patterns; dynamic time warping *(DTW)* [189, 214] does so, but fails to associate series with pattern variations. Since series timestamps reflect how well FLInD compresses them, we propose using them **before** compressing the series to determine whether series can be stored together in FLInD.

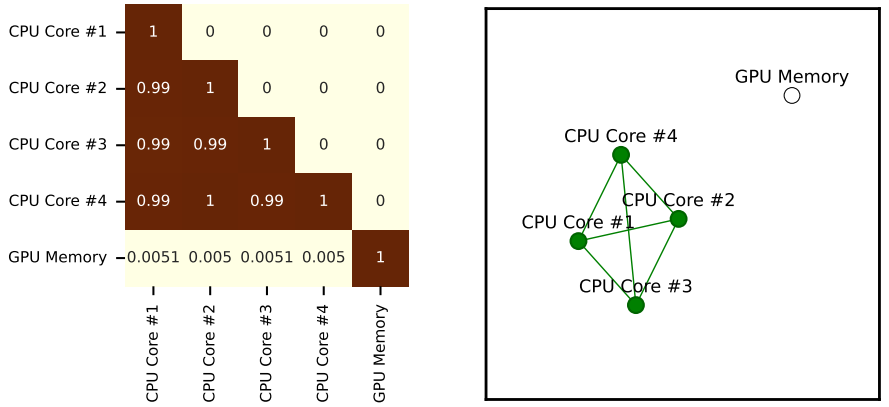
More specifically, when comparing two series, we *(i)* compress each series with FLInD separately in a lossless manner, *(ii)* collect each series timestamps, and *(iii)* compute the timestamps Jaccard index, namely the ratio of all timestamps intersection over all timestamps union. The output quantity, in $[0; 1]$, describes how good FLInD will be in storing both series together. Repeating this process across multiple series, we can build the shared timestamp ratios matrix, and, by exploring it with clustering algorithms, identify groups of well-matched series that provide storage gains when compressed together.

To illustrate how the series cluster search operates, we run it step by step using a system metrics [129] subset, namely four CPU and a GPU memory series. The related timestamps matrix is displayed in Figure 3.14a: it appears that all series will be well compressed by FLInD, except for the GPU memory series. To verify this assumption, we look for the best (size-wise) way to store the aforementioned series by *(i)* putting some in a FLInD instance, and *(ii)* compressing the others using one FLInD instance per series.

[129] Madsen (2022), Hardware Performance Archive [2019]

Please note that this brute-force method is computationally expensive, as there are 2^n ways to split the series storage between FLInD and separate instances (leading to 32 benchmark runs in our example). Detailed results are explored in Appendix A.

Figure 3.14b showcase the matrix conversion to a graph, only creating edges between series sharing timestamps, and looking for the



(a) Matrix of shared timestamps between series.

(b) The biggest clique of the graph, represented in green, regroups all CPU series.

Figure 3.14: Identification of series clusters for compression. Using a value threshold when building the graph of shared timestamps ratios allows us to identify cliques, which hence represent series sharing timestamps.

biggest clique in the graph: here, it indicates all CPU series should be stored in one FLInD instance, and the GPU memory series be stored separately.

Unless said otherwise, the threshold value used to build graphs is **0.1**, *i.e.* associating series that share at least 10% of their timestamps.

RELEVANCE. After integrating multiple datasets to our study, we discovered that our initial hint for series compatibility through the timestamps Jaccard coefficient was not pertinent, leading CLIQUEFLINND to present good results only on the HARDWARE PERFORMANCE PROFILE dataset used during the inception of the algorithm.

In the latter case, CLIQUEFLINND splits the 42 dimensions into 11 groups, providing a storage gain of 33.92% compared to FLInD. With the other datasets, CLIQUEFLINND regroups all series in one single compression group, thus being equivalent to FLInD. Additionally, the compression groups computation time takes up to one hour for the SOLAR POWER dataset.

We subsequently decide to exclude CLIQUEFLINND from our study.

Comparison protocol.

In the majority of our benchmarks, to evaluate the performance of compression approaches, we:

1. select a part of or the complete input dataset;
   1.1) *(optional)* set tolerated interpolation error;
2. load all algorithms listed in Subsection 3.2.4 with the selected dataset;

Table 3.2: Storage budget (in MiB) of compressing datasets; best option is put in bold.

	Hardware performance		Electricity load		Solar Power Data		Microservices Traces	
Raw	39.40		397.00		325.61		143.47	
TimescaleDB	43.05	+ 9%	548.05	+38%	410.76	+26%	∅	
FLI	44.35	+13%	615.64	+55%	271.83	−17%	286.65	+100%
FLInD	39.39	+0%	396.99	+0%	169.90	−48%	143.27	− 1%
FLI△	**15.46**	**−61%** **0.07**	**74.91**	**−81%** **0.067**	**25.46**	**−92%** **0.063**	**58.75**	**−59%** **0.066**
FLInD△	35.71	−9% 0	100.53	−75% 0.071	54.58	−83% 0.073	66.85	−53% 0.068
SplitFLInD△	28.45	−28% 0.005	89.43	−77% 0.073	39.04	−88% 0.08	61.23	−57% 0.074

△: lossy compression. All lossy algorithms were configured to output a comparable MSE of ≈ 0.001. *P*95 compression error values are displayed in gray.

2.1) *(optional)* trigger compression for algorithms requiring the complete series to be collected before compression;

3. Gather metrics from data-loaded instances.

All experiments were implemented in a publicly available notebook [177], and were run on a Dell Latitude 5590 workstation (*i7-8650U* Intel CPU, 32 GB RAM) with Python *v3.12.3*.

3.2.5 *Results*

In this Section, we present experimental results we obtained by storing datasets with the mentioned compression techniques. Table 3.2 lists the cost of storing datasets in memory, using FLInD and the four baselines presented in Subsection 3.2.4; table 3.3 lists associated input/output timings (averaged over 10 iterations).

Despite its time-series-oriented motivation, TimescaleDB does not perform well: its storage cost is consistently higher than raw storage, and it is the slowest of all competitors in read/write operations. In the extreme case of high cardinality series, namely the Microservices dataset with 13,058 dimensions, TimescaleDB is not even suitable, as its underlying SQL tables only support up to 1,600 columns.

Losslessly, linear approximating MTS.

In three of four datasets, FLI performs poorly from a storage budget perspective: it systematically increases storage costs, doubling them in the Microservices traces case. Timestamp duplication between

Table 3.3: Read and write throughput (in operations/second) of storage techniques over several datasets (reads to the left, writes to the right); best option (excluding raw storage baseline) is put in bold.

	Hardware performance		Electricity Load		Solar Power Data		Microservices Traces	
Raw	1,447,618	106,612	722,594	42,892	616,524	39,182	15,687	683
TimescaleDB	130	3,956	34	896	52	1,096	∅	
FLI	7,802	27,772	808	5,205	682	5,363	41	118
FLInD	50,272	**33,220**	**14,132**	13,201	**12,589**	12,673	**424**	344
FLI△	9,481	26,443	1,004	5,486	932	5,098	37	136
FLInD△	**50,369**	32,594	14,013	**13,777**	12,512	**12,783**	420	**351**
SplitFLInD△	39,829	20,542	8,578	5,298	7,613	4,353	250	129

△: lossy compression. All lossy algorithms were configured to output a comparable MSE of ≈ 0.001.

independent instances negatively affects the storage budget. Additionally, FLI is also bad in reads/writes per second, as it is second-to-last only to TimescaleDB.

On the other hand, FLInD consistently reduces storage size across all datasets compared to the raw baseline. Indeed, since it stores all dataset points at worst, it cannot perform worse than raw storage. The best situation for FLInD is when all dimensions of a MTS evolve linearly, whether consecutive values are the same or linearly increase/decrease, leading FLInD to skip points thus providing a memory gain: this happens in the *Solar Power Data* dataset, whose solar panels all output a production of 0*W* during night time; within this dataset, 53% of consecutive values deltas have a value of zero, leading to FLInD providing a 48% size gain.

In all other datasets, FLInD has the same budget as raw storage: the observed limited size gain compared to raw storage is due to FLInD skipping data points when all dimensions do not break their invariants; the more dimensions, the less likely this is.

> **RQ 1**: This result shows that the linear approximation is relevant for storing MTS; in particular, FLInD is relevant, providing a gain on datasets containing linearly evolving data and performing as well as raw storage in the 0-error worst case.

Lossy compression of time series.

As described in Section 3.2, FLInD allows error tolerance when compressing data, trading data accuracy for an improved storage gain. Tables 3.2 and 3.3 display the performances of both Fast Linear Interpolation (FLI)'s and FLInD's lossy versions, each applying *an error*

tolerance of one standard deviation per series.

Storage-cost-wise, lossy FLI is the best competitor with all datasets, providing a storage gain of up to 92% in the *solar power* dataset; however, this has consequences, as the price to pay here is an error when compressing data, as shown by the 95th percentile ($P95$) value of 0.063.

Regarding input and output throughput, FLInD keeps the lead, with similar performance regardless of its lossiness: it is the fastest competitor in both READ and WRITE operations across all datasets.

Distributing series between several instances.

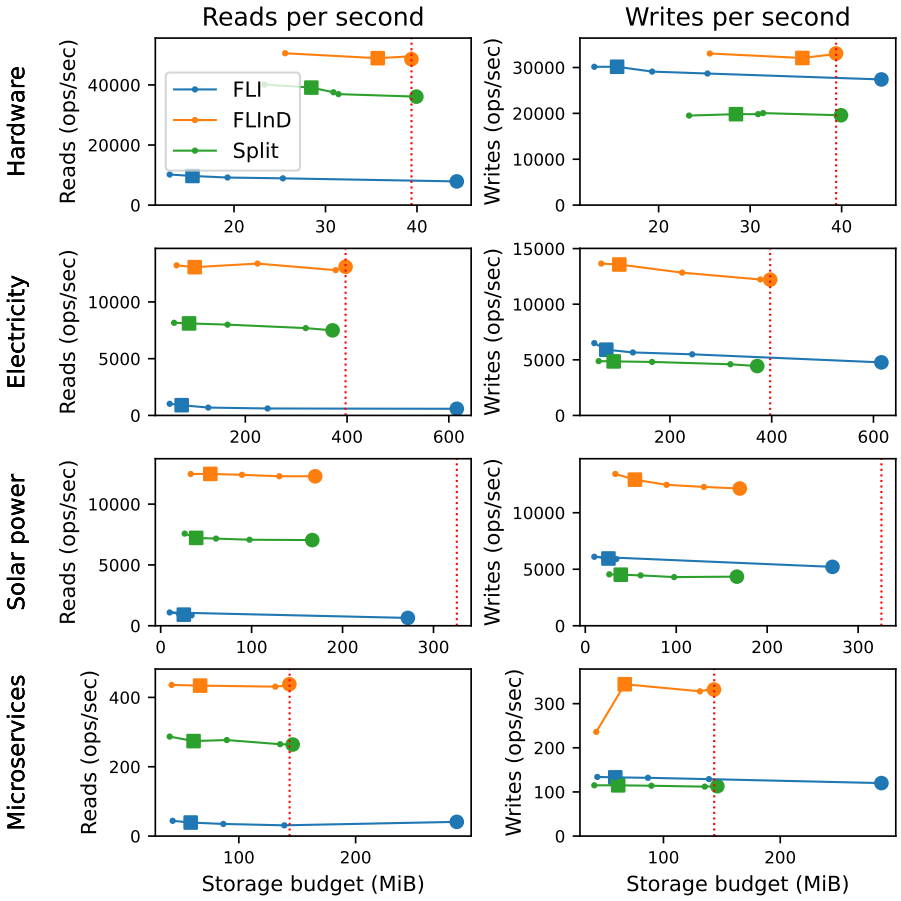


Figure 3.15: Read and write throughput (in operations/second) of linear approximation competitors over several datasets, one per line. The first column lists read operations per second; the second, write operations per second.
The red dotted line marks the baseline for the raw storage budget. For each technique, a circle marker marks the lossless variant, and a square marker marks a lossy variant configured to output a comparable MSE of ≈ 0.001.
Smaller points additionally present intermediate configurations of competitors with a varying error.

Table 3.4: Performances metrics over compressing the Microservices Traces dataset using an increasingly smaller count of dimensions per instance. The lower the dimensions count, the better the error is acknowledged, leading to better total size, but also to increased read/write times due to the dataset being split between many FLInD instances.

Name	Size (MiB) ↓	Reads/sec ↓	Writes/sec ↓	*P*95 error ↑
FLInD	143.27	380	257	0.0
100 series per instance	72.63	260	177	0.030
50 series per instance	59.76	181	99	0.038
20 series per instance	44.37	115	48	0.049
10 series per instance	35.05	95	29	0.057
5 series per instance	27.77	59	17	0.066
2 series per instance	21.27	29	7	0.078
1 series per instance (FLI)	18.44	17	4	0.088

Figure 3.15 sums up performances of the FLI (in blue) and FLInD (in orange) approximation models. Square points show for each algorithm I/O throughput associated with an observed compression mean squared error of 0.001: FLInD is better than FLI in both reads and writes per second, but presents a slightly increased storage budget.

Since the error tolerance of FLInD is counteracted by the increasing count of dimensions of a series, gaining storage space can be done by reducing the number of stored dimensions per FLInD instance.

Dividing a series' dimensions between multiple FLInD instances covers a configuration space that allows for picking a trade-off between minimised storage budget and increased I/O performances, the minimal bound being storing all series together — which means using FLInD as is — and the maximum bound being using as many FLInD instances as there are dimensions to store, compressing one series per instance (which resolves to using FLI).

Table 3.4 shows the results obtained when varying the number of FLInD instances on the Microservices Traces dataset, ranging from one single overall instance for all series (vanilla FLInD), to 1 series per instance (FLI). We observe that splitting series across multiple instances affects their storage budget, yielding a 87% size gain in the extreme one-instance-per-series case compared to using FLInD alone. However, the more the data is split across multiple instances, the more degraded access and write performance become: algorithms must perform additional logic during read (gathering all series across all instances) and write (splitting data point series across instances) operations.

Based on this observation, we arbitrarily chose the value of 50 series per FLInD instance in the SplitFLInD competitor (50 being less than any count of dimensions of any dataset used in this work). As

expected, SplitFLInD allows navigating the dataset storage cost and compression error *VS* read/write throughput. With all four datasets, SplitFLInD is between FLI and FLInD in terms of storage budget, requiring fewer bytes than FLInD but requiring more than FLI, in exchange for a limited compression error observed on data: in the *hardware profile* dataset case, it even provides a $p95$ value of 0.005, as opposed to 0.07 for FLI.

Regarding input/output performance, SplitFLInD is again between FLI and FLInD in terms of writes per second, second behind FLInD (lossless or not); however, it falls short of FLI in read throughput, sacrificing read speed for reduced data degradation.

> **RQ 2**: Results show that the series can be compressed even further when tolerating an error during compression, but at the cost of increased read/write speed: a trade-off must be made between instance size and I/O performance.

• • •

The two previous contributions focused on storing temporal data on smartphones: Section 3.1 introduced Fast Linear Interpolation (FLI), leveraging linear regression to allow big data compression within constrained storage budgets, and Section 3.2 presented Fast Linear Interpolation in n-Dimensions (FLInD), a FLI extension allowing compression of multi-dimensional series.
We will now see how this storage component is integrated into the IN-siTu locAtion proteCTion (INTACT) framework, a mobile privacy system allowing the execution of privacy attacks and privacy protection algorithms directly on smartphones.

3.3 PROTECTING USER PRIVACY THROUGH LOCAL DATA STORAGE

The INTACT framework is a mobile (developed with Dart, deployed on Android and iOS) privacy system allowing the execution of privacy attacks and privacy protection algorithms directly on smartphones. Leveraging the FLI storage component, INTACT's *attack/protect* loop allows end users to ensure their data is safe **before** sharing it with third-parties.

This section subsequently delves into INTACT's privacy protection element, introducing *(i)* a new way to locally compute POIs and *(ii)* an *in situ* Location Privacy Protection Mechanism (LPPM).

3.3.1 *Divide&Stay: Attacking Location Privacy*

After considering the problem of time series storage in the previous section, we now present our mobile-ready geolocation privacy attack tool: Divide&Stay (D&S). The purpose of such a tool is to audit the privacy of mobility traces directly where they are generated: on resource-constrained devices such as smartphones. More specifically, this audit consists in locally attacking mobility traces (potentially after applying protective mechanisms to them) to assess the risk involved in sharing said location data. Only then would it be reasonable, privacy-wise, to upload one's location traces to a third party, or deciding not to upload anything if the risk is deemed too high.

[169] Primault et al. (2014), "Differentially private location privacy in practice"

D&S constitutes an enhancement of the original POI-Attack algorithm proposed by Primault et al. [169]. Because the original contribution is too costly to run on mobile devices, D&S proposes a mobile-ready variant.

In POI-Attack, the POI disclosure is done by a two-steps algorithm: potential candidates for POIs (dubbed *stays*) are first extracted, then *stays* are merged to avoid duplication of similar POIs. A *stay* is defined as a circle with a radius lower than d_{max} where a user spent a time higher than a set time t_{min}. A *stay* is represented by its centre. The two thresholds t_{min} and d_{max} have an important impact on the type of POI extracted. *Short stays* will identify day-to-day patterns, such as shopping preferences, while *long stays* will identify *e.g.*, travel preferences. In the second step, the *stays* whose centroids are close enough are merged to obtain the final list of Point Of Interest. POI-Attack [169] iterates linearly over the mobility trace and compute *stays* as they appear. This approach is expensive for denser mobility traces—*i.e.*, high-frequency sampling. It is prohibitively long to execute on constrained devices like mobile phones.

Our contribution Divide&Stay (D&S) instead proposes a divide-and-conquer strategy that scales with the data density. The intuition behind D&S is to avoid wasting time looking for *stays* in portions of the trace that are highly unlikely to contribute to a stay, *i.e.* where more than d_{max} has been travelled in less than t_{min}, *e.g.*, a car trip at high speed in a straight line. While the regular approach would consider each location until the end of the trace, D&S skips it entirely. The key idea of D&S is to recursively divide the trace until either such a *stay*-less segment is found and discarded, or until a fixed-size segment is found on which the regular way to extract *stays* is performed.

Algorithm 4 depicts the pseudo-code of Divide&Stay. It scrutinizes the GPS trace $T \in (\mathbb{R} \times \mathbb{G})^n$, composed of n samples. For each $i \in [0, n-1]$, $T[i].t$ represents the i_{th} sample's timestamp, while $T[i].g$ is its position in whatever geographical space $\mathbb{G}$ equipped with a distance function $\text{dist}_{\mathbb{G}}$. D&S has three configuration parameters: the aforementioned t_{min} and d_{max} representing the time and space limits

Algorithm 4 Divide&Stay using parameters $(t_{min}, d_{max}, s_{max}) \in \mathbb{R}^{3+}$

Require: $T \in (\mathbb{R} \times \mathbb{G})^n$

1: **function** DIVIDE&STAY($(i_{\text{first}}, i_{\text{last}}) \in [0, n-1]^2$)
2: **if** $i_{\text{last}} - i_{\text{first}} \leq s_{max}$ **then** ▷ Iterative case
3: **return** getStays $(T[i_{\text{first}}..i_{\text{last}}])$
4: **end if**
5: $S \leftarrow \emptyset$
6: $i_{\text{split}} \leftarrow \lfloor (i_{\text{first}} + i_{\text{last}})/2 \rfloor$
▷ Left sub-trace recursion
7: $t_\Delta \leftarrow T[i_{\text{split}}].t - T[i_{\text{first}}].t$
8: $d_\Delta \leftarrow \text{dist}_{\mathbb{G}}(T[i_{\text{first}}].g, T[i_{\text{split}}].g)$
9: **if** $\neg(d_\Delta > d_{max} \wedge t_\Delta \leq t_{min})$ **then**
10: $S \leftarrow$ Divide&Stay $(i_{\text{first}}, i_{\text{split}})$
11: **end if**
▷ Right sub-trace recursion
12: $t_\Delta \leftarrow T[i_{\text{last}}].t - T[i_{\text{split}}].t$
13: $d_\Delta \leftarrow \text{dist}_{\mathbb{G}}(T[i_{\text{split}}].g, T[i_{\text{last}}].g)$
14: **if** $\neg(d_\Delta > d_{max} \wedge t_\Delta \leq t_{min})$ **then**
15: $S \leftarrow S \cup$ Divide&Stay $(i_{\text{split}}, i_{\text{last}})$
16: **end if**
17: **return** S
18: **end function**

of a POI, and s_{max}, the sub-trace size threshold below which the divide-and-conquer approach for POI inference is abandoned in favour of the iterative one. Three indices are manipulated, all called i with a self-explanatory subscript. The Divide&Stay function takes i_{first} and i_{last} as arguments, being the bounds of the sub-trace under study. On the first call, the whole input space is provided: i_{first} is 0 and i_{last} takes $n-1$. Subsequent recursive calls provide either the first or second half of the input sub-trace until an iterative search is preferred.

On lines 2 to 4, the size of the input sub-trace is checked against the threshold s_{max}. If the trace is smaller, then a linear search for *stays* is performed *à la* POI-ATTACK [169]. On l. 6, the indices space is split: i_{split} is set to the midpoint between i_{first} and i_{last}. Lines 7-11 check whether the left sub-trace $T[i_{\text{first}}..i_{\text{split}}]$ is susceptible to contain *stays*, in which case Divide&Stay is recursively called. Its output fills the list of stays S. As already mentioned, a sub-trace cannot contain any *stay* if a distance of more than d_{max} was travelled in less than t_{min}. Lines 11 to 16 perform the same check for the right sub-trace $T[i_{\text{split}}..i_{\text{last}}]$, in which case the result of the recursive call is added to S. Finally, S is returned. POI-ATTACK's merge of *stays* into POIs must be subsequently performed.

The more discarded segments, the faster compared to the regular approach. *Stays* around the midpoints i_{split} could be missed, but D&S ignores them because a POI is a cluster of several stays: It is very

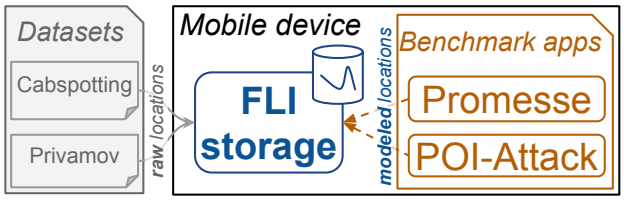


Figure 3.16: Architectural diagram of our *in-situ* big data processing experiment on a mobile device.

unlikely to miss any of them. D&S can be implemented sequentially or concurrently, to leverage multi-core processors.

3.3.2 *Enabling* in-situ *big data processing*

Location data is not only highly sensitive from a privacy perspective but also crucial for location-based services; as such, it can be protected by Location Privacy Protection Mechanisms (LPPMs), ensuring the privacy/utility trade-off is met before sharing data with third parties. While LPPMs have been developed to protect user locations, they are generally used on the server where the data is aggregated. The user location data is thus exposed to classical threats, such as malicious users, man-in-the-middle, or database leaks.

To avoid such threats, one privacy-preserving solution would be to keep the data on the device where it is produced until it is sufficiently obfuscated for sharing with a third party. With GPS data, this protection mechanism must be undertaken by a device-local LPPM. Evaluating the privacy of the resulting trace must also be performed locally, by executing attacks on the obfuscated data. Both processes require storing all the user mobility traces in the mobile device.

[109] Khalfoun et al. (2021), "EDEN: Enforcing Location Privacy through Re-identification Risk Assessment: A Federated Learning Approach"

While existing approaches have simulated this approach [109], no real deployment has ever been reported. In this Section, we show that using FLI enables overcoming one of the memory hurdles of constrained devices. We use FLI to store entire GPS traces in mobile devices, execute POI attacks with D&S, and protect the traces using the LPPM Promesse.

Promesse [170] is an LPPM that intends to hide POIs from a mobility trace by introducing a negligible spatial error. To do so, Promesse smooths the trajectories by replacing the mobility trace with a new one, applying a constant speed while keeping the same starting and ending timestamps. The new trace T' is characterised by the distance δ between two points. First, additional locations are inserted in chronological order based on the existing locations. If the distance between the last generated location $T'[i]$ and the current one $T[c]$ is below δ, this location is discarded. Otherwise, $T'[i+1]$ is not defined as the current location $T[c]$, but the location between $T'[i]$ and $T[c]$, such that

the distance between $T'[i]$ and $T'[i+1]$ is equal to δ. Once all locations in the new mobility trace are defined, the timestamps are updated to maintain the time interval between the two locations, while keeping the timestamps of the first and last locations unchanged. The resulting mobility trace is protected against POI attacks while providing high spatial accuracy.

We therefore implemented mobile versions of POI attacks and PROMESSE [170] using the Flutter library of FLI and evaluated their performance on the PRIVAMOV dataset. Fig. 3.16 depicts the resulting experimental deployments of POI attacks and the corresponding LPPM in mobile devices, alongside FLI. Our experiments are performed using a publicly available application [173].

[173] Raes et al. (2022), In-situ LPPM

Key metrics.

POI QUALITY.
Since we wish to assess the effectiveness of the protection mechanism, we need to evaluate the extent to which the attack carried out by D&S is successful, *e.g.*, whether it infers POIs naturally present in the trace.

Measuring the quality of inferred POIs is difficult, as there is no acknowledged definition of how to compute POIs. We consider as our ground truth the POIs inferred by the state-of-the-art POI-attack [169], which we refer to as the 'raw' POIs. The existence of such a 'ground-truth' is, however, debatable, as two different—but close—POIs can be merged by the algorithm into a single POI. As an example, if a user visits two different shops separated by a road, but their distance is lower than d_{max}, those will be merged into a single POI located at the centre of the road. For that reason, we need two metrics to compare the sets of POIs returned in the different cases: the distance between POIs, and the sets' sizes.

Distance between POI. As the POI definition is mainly algorithmic, we compute the distance of each obtained POI to its closest raw POI as the metric assessing the quality of the new POIs. These distances are reported as a CDF. If FLI does not significantly alter the locations of the mobility, the traces it captures, and the computed distances should be short.

Number of POIs. In addition to the distances between POIs, we are also considering the quantity of returned items as a metric. In our previous example, visiting the two shops may result in two different POIs because they have been slightly shifted by FLI. Beyond the numbers, we expect that PROMESSE successfully anonymises mobility traces by returning a total of zero POI.

Table 3.5: Execution time for PRIVAMOV `user 1` on different mobile platforms. Previous deployments of these algorithms were made on desktop computers (as reference, POI-ATTACK execution takes one hour to complete on a laptop). FLI now enables its instantiation on ubiquitous devices, with reasonable processing times.

Task	Moto Z	Fair Phone 3	Google Pixel 7 Pro	iPhone 12	iPhone 14+
PROMESSE	1.5*s*	1.3*s*	0.4*s*	0.2*s*	0.2*s*
D&S	114.4*s*	109.2*s*	30*s*	18.8*s*	19.4*s*

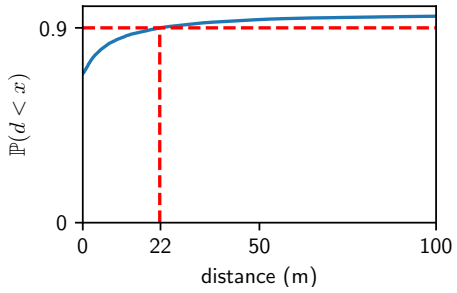


Figure 3.17: Distances distribution when using D&S on CABSPOTTING. The distances between the POIs are obtained using D&S and their closest counterparts, obtained with the traditional POI attack. Except for a few extreme values, the values are close: more than 68% are the same and 90% of the POIs are at a distance lower than 22 meters from a "real" one.

Results.

Given that PRIVAMOV is the largest dataset, Table 3.5 reports on the worst-case processing times observed for `user 1` on different mobile platforms. This captures a user mobility trace of 4,341,716 locations reported by a mobile device. It shows that applying PROMESSE to the `user 1` mobility trace takes, on average, 1 second to run across all devices. More interestingly, POI-ATTACK takes no longer than 30 seconds ($\pm$ 43*ms*) to run on the latest generation hardware, both on Android and iOS. This is a significant improvement over the state-of-the-art implementation [169], which takes 1 hour to run on a desktop computer. We believe that this is a critical step forward towards improving user privacy, as all LPPM experiments to date have been either simulated or centralised, contrary to what the literature suggests [124].

[124] Luxey et al. (2018), "Sprinkler: A probabilistic dissemination protocol to provide fluid user interaction in multi-device ecosystems"

In addition to speed, the quality of the inferred POIs is the most salient concern about D&S. We assess the quality by computing the distances to the POIs obtained from the POI-attack on CABSPOTTING. We choose CABSPOTTING because computing it on PRIVAMOV is prohibitive in terms of computation time. Fig. 3.17 depicts the distribution of the distances below 100 meters: more than 68% are the same, and 90% of the POIs are at a distance lower than 22 meters from actual ones. Fig. 3.18 shows some of the POI inferred from raw samples and the

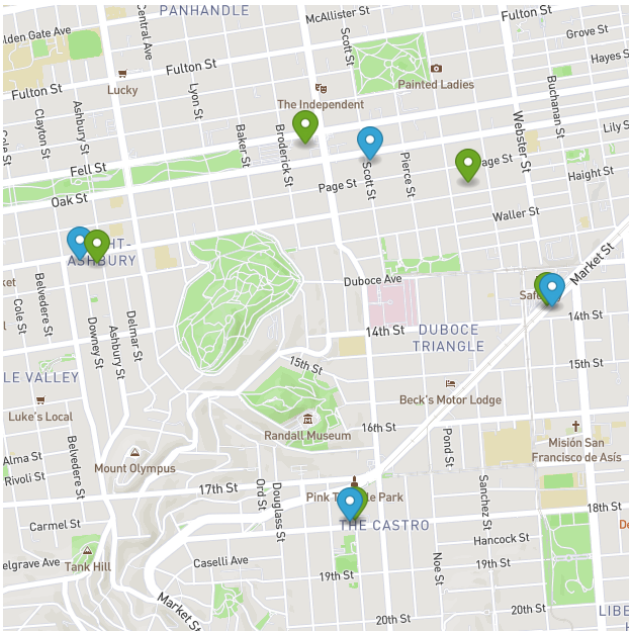


Figure 3.18: Example of POI of `user 0` from CABSPOTTING using the standard POI algorithm on raw data (in **blue**) and those obtained on the data modelled by FLI and computed with Divide&Stay (in **green**). The inferred POIs are very close on average, but there are some outliers.

Table 3.6: Impact of FLI and Divide&Stay on the number of inferred POIs from `user 0` trace in CABSPOTTING. Thanks to FLI and Divide&Stay (D&S), PROMESSE protects user privacy at the edge.

	without PROMESSE		with PROMESSE	
Algorithm	**Raw POIs**	**FLI**	**Raw POIs**	**FLI**
POI-attack	30	31	0	0
Divide&Stay	30	30	0	0
POI-attack ∩ Divide&Stay	21	20	-	-

standard POI-attack (in blue) and POI obtained with FLI and D&S: they are different but very close, highlighting the negligible impact for applications that depend on location data.

Finally, Table 3.6 highlights that PROMESSE works well independently of the use of FLI or D&S: it successfully hides all the POI. Therefore, Divide&Stay (D&S) provides an important speed-up of the privacy audit process, reducing the order of magnitude from the hour down to the minute, without altering the quality of POIs. Note that FLI was not used in this case, as the performances of D&S are orthogonal to the use of a temporal database to model the samples.

To conclude, our implementation of the data stream storage solution, FLI, enables the effective deployment of more advanced techniques, such as EDEN [109] or HMC [130]. This may require new algorithms,

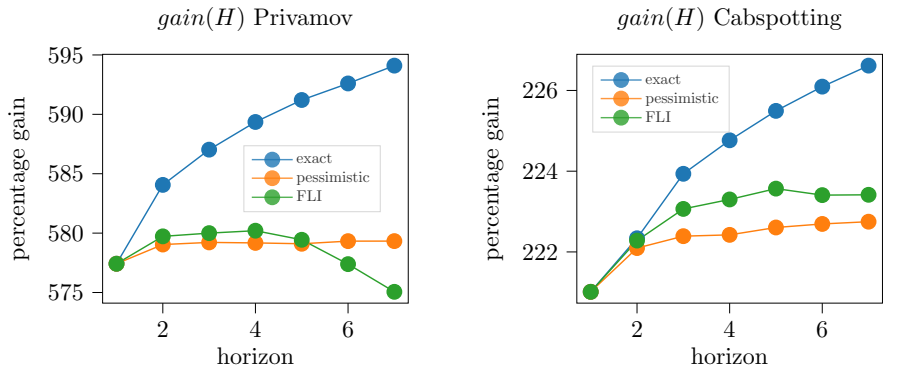
such as D&S, but it enables *in situ* data privacy protection before sharing any sensitive information. We believe this is a critical step forward in improving user privacy, as all LPPM experiments to date have been either centralised or simulated.

Thanks to its FLI storage layer and D&S attack layer, INTACT improves privacy, and the cost of this improved privacy is *in situ* data computation, which requires running potentially heavy data tasks on constrained devices. We report on the execution time of POI-Attack in Table 3.5, and argue that computation time is the price to pay for increased privacy.

• • •

We have seen how, once integrated into the INTACT framework, FLI increases users' privacy by enabling local computation of personal data to avoid data sharing, thereby reducing privacy leaks. We will now discover how, as a linear regression technique, FLI can also be used to detect privacy leaks before they occur.

3.4 APPLICATION OF A PREDICTIVE METHOD TO PROTECT THE PRIVACY OF MOBILITY DATA



(a) Privacy gain against Privamov users.

(b) Privacy gain against Cabspotting users.

Figure 3.19: Average privacy gain against real data using all dataset users.

[145] Molina et al. (2023), "Optimal privacy protection of mobility data: a predictive approach"

Subsection 2.1.3 previously explained how Location Privacy Protection Mechanisms (LPPMs) can be used to transform data in both privacy-protecting and utility-preserving ways. Molina et al. [145] introduced MPC-*H*, a Model Predictive Control (MPC) based approach that considers *H* future locations in obfuscating data to be transferred to LBSs: they show using a mobility predictor allows for increased privacy protection, with a privacy gain of up to 17% compared to Geo-I [10], said privacy gain increasing with the *H* size of the prediction horizon. This contribution assumes knowledge of users' future

locations, which is valid in an offline setting but not in a real-world online setting.

To tackle this issue, we [147] subsequently propose to integrate mobility predictors into the system to dynamically determine the noise to be added to location data before transmission. Since it relies on linear regression, FLI can be used as a mobility predictor when storing location data: its latest piecewise function can be extended to allow interpolation for timestamps beyond the current time (*"in the future"*).

[147] Molina et al. (2025), "Application of a predictive method to protect privacy of mobility data"

Two other predictors are implemented in this work. The first one is an oracle that predicts the exact future locations; although unrealistic, it can serve as a baseline to quantify the maximum achievable privacy gain, facilitating comparisons with other mobility predictors. The second one is a *worst-case* predictor, simulating a user who visits all locations at every instant to minimise the privacy gain.

Figure 3.19 displays privacy gain offered by the MPC based approach compared to using raw location data, deployed on two location datasets, PrivaMov [144] and Cabspotting [163]. It shows that, as expected, the privacy gain offered by MPC-H increases with the H size of the prediction horizon, when using the oracle exact predictor. Despite its linear fashion directly assuming users either move in straight directions (no turns) at constant speeds or stay in place without moving, FLI still provides an increased privacy gain for short-term horizons with PrivaMov; it however performs worse than the pessimistic predictor for H values greater than 5, which was also expected, as user linear movement probability increases with the size of the horizon. The privacy gain observed for Cabspotting is lower, as this dataset contains only users who move by car, which reduces privacy risks and thus the potential privacy gain.

We showed here that using linear estimators to predict mobility data for small time horizons provides a privacy gain compared to state-of-the-art LPPMs; additionally, FLI provides an execution time lower than the time between two location transmissions (30 seconds) and has been implemented on Android and iOS smartphones [181], showing a real world deployment of MPC-H is possible.

3.5 DISCUSSION

3.5.1 *Threats to validity*

We followed best development practices, such as documentation and unit testing, during the experimental phase of our studies; however, our implementations of FLI, FLInD, and D&S may contain software bugs that affect the reported performance, hindering our results. To limit this threat, we make the code for our libraries and applications freely available (all artefacts are listed in Section 1.4) to encourage

the reproducibility of our results and to share the implementation decisions we made.

Regarding the implementation of INTACT, hardware threats relate to the constrained device classes we considered. In particular, we focused on smartphones, the most commonly deployed mobile devices in the wild. To limit bias introduced by a given hardware configuration, we deployed FLI on both recent Android and iOS smartphones for most of the reported experiments, while also considering the impact of hardware configurations on reported performance.

Another potential bias concerns the mobility datasets we considered in this contribution. To limit this threat, we evaluated our solutions on two established mobility datasets, CABSPOTTING and PRIVAMOV, which differ in their characteristics. Yet, we could further explore the impact of these characteristics (sampling frequency, number of participants, duration, and scales of the mobility traces). Beyond mobility datasets, we could evaluate other IoT data streams, such as air quality metrics, to assess FLI's ability to handle a wide range of data. To mitigate this threat, we reported on the storage of timestamps, acceleration data, heartbeat data, and 2-dimensional locations.

Additionally, our results might depend strongly on the parameters we choose to evaluate our contributions. While FLI performances (gain, memory footprint) vary depending on the value of the ϵ parameter, we considered a sensitive analysis of this parameter, and we propose a default value $\epsilon = 10^{-3}$ that delivers a minimum memory gain that limits the compressing error.

As a generalisation of the FLI technique, we also view FLInD as a middleware for devices other than smartphones; we implemented it in Python. An interactive notebook [177] allows readers to better understand how FLInD works, try to load FLInD instances with various datasets, and reproduce the figures presented in this work.

All compression competitors were implemented in this notebook, thus relying on the performance of a Python interpreter. Despite using Numpy [158] arrays and associated operations, the comparison with other techniques might be unfair: for instance, during evaluation, the read/write workload for TSDBs occurs in Docker containers, not in the Python client code. A way to address this potential issue could be to implement all competitors consistently, for instance, in C.

[158] Oliphant (2006), NumPy website

3.5.2 *Future works*

ERROR TOLERANCE. Regarding the error tolerance applied to lossy algorithms, we *de facto* used one standard deviation per series. Other indicators such as average values could also be provided to FLInD and produce better results.

FLI and FLInD currently take the $\vec{E}$ list of tolerated errors as input, meaning developers must know the data that will be compressed by the algorithms to enter adequate values. Using an input buffer, FLInD could analyse the data and automatically compute the optimal error tolerance before compressing it.

Having an input buffer could also enable FLI and FLInD to detect whether a series is derived from another (shifted patterns, pattern variations), and to store related pattern information to further reduce the final storage cost of the complete series.

TIME-VARYING COMPRESSION. Although FLI and FLInD increase storage capacity through data modelling, they might still reach the storage limit of their host device if using a constant ϵ parameter (which drives the compression rate). To address this issue, we could dynamically adapt data compression to fit a storage size constraint. Toward this end, An *et al.* [9] propose an interesting time-aware adaptive compression rate, based on the claim that data importance varies with its age.

[9] An et al. (2022), "TVStore: Automatically Bounding Time Series Storage via Time-Varying Compression"

NATIVE IMPLEMENTATION. Beyond the current implementation reported in Section 3.3, one could envision a native integration of INTACT into Android and iOS to enable LPPMs for any legacy application. This technical challenge mostly consists of packaging FLI and Divide&Stay (D&S) as a new `LocationProvider` in Android[65] and a new `CLLocationManager` in iOS [15]. Interestingly, such a native integration of INTACT can allow end users to configure the list of enabled LPPMs and their related settings through the operating system control panel.

[65] Google (2011), Android `LocationManager` documentation

[15] Apple (2013), iOS `CLLocationManager` documentation

Although not implemented in practice, INTACT includes a GPS provider part, made of LPPM protections and POI attacks, which can supply obfuscated locations to LBS apps; This, however, is straightforward to implement on mobile devices using system abstractions available in both Android [65] and iOS [15].

MORE LPPMS. The increased storage capacity offered by FLI not only allows for increased mobility data storage, but also allows applying *in-situ* LPPMs requiring lots of data to work, for instance, those offering k and l-anonymity guarantees by hiding the user among others [127, 200].

3.5.3 *Perspectives*

Beyond the deployment of LPPM at the edge of the network, we believe that our contributions open new avenues for improving user privacy and, more generally, enabling data processing at the edge. This includes implementing alternative LPPMs, implementing differential

privacy algorithms such as *k-anonymity* and *l-diversity*, and including additional privacy attacks to provide users with privacy reports for the data they intend to share.

FLI demonstrates that increasing the storage capacity of data streams on constrained devices offers new distributed computation models, inspired by the big data principles, that can deploy automated processing tasks on edge devices. Beyond LPPMs, one can consider using mobile devices in a federated or decentralised learning environment to train models without compromising sensitive personal information. Furthermore, we demonstrated with Divide&Stay (D&S) that one can execute privacy attacks at the scale of a device, thereby enabling the assessment of a model's sensitivity before sharing it with a third party.

While the literature in mobile privacy is rich, we hope that open-source implementations of our libraries will encourage the research and developer communities to embed our contributions to strengthen their users' privacy. We believe that FLI, FLInD, and D&S are key software assets that can stimulate the mobile database and privacy communities to deliver effective solutions on many devices, including mobile devices, which are widely deployed nowadays.

3.5.4 *Conclusion*

Modern IT devices (servers, laptops, smartphones) produce huge amounts of data streams, which are often forwarded to remote third-party services for storage and processing. This data processing pattern may lead to privacy breaches, as the raw data may leak sensitive personal information. Furthermore, the volume of data to be processed may require substantial storage capacity across mobile devices and remote servers, as well as network capacity to handle the increasing number of devices deployed in the wild.

On the other hand, while LPPMs are a promising solution to protect user locations, they are not widely deployed in practice, mostly because of their high computational cost, which is prohibitive on mobile devices.

We therefore propose INTACT, a software framework that enables LPPMs on mobile devices by leveraging their increased storage capacity.

The contributions of INTACT are threefold: we leverage *i)* a compact storage system based on a piece-wise linear model dubbed FLI (proposed in [180]), *ii)* introduce a new way to compute POIs, called D&S, and finally *iii)* demonstrate how FLI could unlock device-local privacy protections on time series while using machine learning.

Additionally, we propose FLInD, an optionally lossy storage middleware for multivariate time series that leverages a piece-wise linear model to store as many dimensions as needed (tested up to 13,058 in this work). This algorithm improves on FLI by guaranteeing a

non-degraded storage cost compared to storing raw series, providing greater storage gains on co-linear datasets, and ensuring high read/write throughput, even in a lossy setting.

From a perspective, we believe that FLI can serve as a building block for the development of privacy-friendly mobile applications that store and process data locally, without relying on third-party remote services. In particular, one can explore the deployment of federated learning algorithms on mobile devices to train machine learning models without compromising sensitive personal information. In the long term, FLI paves the way for implementing decentralised machine learning algorithms on mobile devices by leveraging device-to-device communication models, such as SPRINKLER [124]. For example, we believe Sensitive Personal Information (SPI) captured by ubiquitous devices should be anonymised locally *before* any data exchange.

While FLI can store large amounts of data on mobile devices, D&S provides a significant speed-up, reducing the total computation time of POI attacks by several orders of magnitude, making them suitable for mobile computing.

By sharing this INTACT framework with mobile developers, we make an important step forward towards the real deployment of LPPMs and, more generally, privacy-friendly data-intensive workloads at the edge (*e.g.*, federated learning on mobile phones).

4

LOCAL DEVICE-TO-DEVICE COMMUNICATION

We have seen in the previous chapter how local storage provides greater privacy against Location-based service (LBS) providers by transforming data before sending it to cloud actors, preventing private data extraction. Another straightforward way to avoid cloud accessing user data is to avoid using the cloud at all. This chapter thus delves into another issue: how can peers in the same vicinity communicate without going through the Internet?

This chapter explores Device-to-device (D2D), a communication paradigm that relies on peer-to-peer, local (*i.e.*, without Internet) communication between users. Section 4.1 presents a listing of existing technologies supporting the deployment of D2D, depicts its security properties, then discusses why D2D communication is not widespread; Section 4.2 subsequently introduces Venice, an multiplatform implementation of the D2D paradigm.

Contents

4.1 An attractive, yet poorly adopted communication means **66**
4.1.1 Communication channels 66
4.1.2 Security models 73
4.1.3 Software Hurdles 77
4.1.4 The discouraging D2D user experience . . 80
4.2 A communication framework to remove usage friction **83**
4.2.1 Connection process. 83
4.2.2 A device-to-device communication framework 84
4.3 Discussion **86**
4.3.1 Threats to validity 86
4.3.2 Future works 87
4.3.3 Conclusion 88

4.1 AN ATTRACTIVE, YET POORLY ADOPTED COMMUNICATION MEANS

This Section introduces the current deployment state of the Device-to-device (D2D) paradigm, presenting different communication channels (such as Bluetooth or Wi-Fi) that can be used in its implementation, as well as interesting security properties it guarantees. Then follows a discussion over limits preventing its adoption, supported by a survey.

4.1.1 *Communication channels*

The *Radio Frequency (RF)* spectrum is regulated in every country: it is forbidden to emit on specific frequency bands without authorisation. Consequently, each RF channel resides on an agreed-upon frequency band.

This subsection introduces both *out-band* (unlicensed) channels and *licensed* cellular networks, then presents several proprietary solutions leveraging physical channels in order to provide user-friendly D2D connectivity; finally, it concludes by explaining how all channels are made available to the users on the hardware level.

Table 4.1 presents a compilation of communication channels considered the most readily available for D2D applications to date —interestingly, all are out-of-band. Channels are organised into families either because they share a similar spectrum or because they are part of the same family of standards. For each standard's family, several specifications were displayed to show the evolution of their characteristics, as specification efforts progressed.

4.1.1.1 *Out-band channels*

FREE-SPACE OPTICAL COMMUNICATION (FSO)

Free-space optical communication has been used by humanity since its infancy — from smoke signals to optical telegraphs to the most recent visual light communication technologies.

INFRARED COMMUNICATION. Infrared refers to the electromagnetic spectrum between 300 GHz and 430 THz (visible red light), and was first 'discovered' in 1800 by astronomer Herschel [182]. Since then, it has found countless applications, from night vision to astronomy to communication. As a consumer product, infrared is routinely used, e.g., in a television's remote control. Nintendo featured infrared sensors in most of its gaming consoles, from the 1998 Game Boy Colour to the 2017 Switch [157]. It seems, however, that only a few games ever used the feature, highlighting the developer's preference for other wireless communication channels.

[182] *Rowan-Robinson (2012)*, Night Vision

[157] *Nintendo Wiki (2022)*, List of Nintendo Systems

Nevertheless, to this day, several smartphone manufacturers continue to ship infrared 'blasters' in their products, Xiaomi notably pro-

Table 4.1: Characteristics of the most widely available D2D communication channels

Name	Specification date	Device discovery *	Maximum distance **	Maximum data rate	Frequency
Optical					
Infrared (IrDA)	1994 [100]	Manual	Several meters	16 Mb/s	0.3-430 THz
QR Code-flashing	—	Manual	Around a meter	23.6 kb/picture	440-790 THz
Wi-Fi					
802.11 (legacy)	1997 [88]	Automatic	100 m	1-2 Mb/s	2.4 GHz
802.11ax (Wi-Fi 6E)	2021 [92]	Automatic	100 m	1.2 Gb/s	1-7.125 GHz
802.11ay	2021 [93]	Automatic	10 m	100 Gb/s	60 GHz
Bluetooth					
Bluetooth Core v1	1999 [27]	Automatic	100 m	1 Mb/s	2.4 GHz
Bluetooth Core v5	2016 [29]	Automatic	100 m	1-3 Mb/s [30]	2.4 GHz
RFID					
NFC	2004 [95]	Manual	20 cm	106-424 kb/s	13.56 MHz
Ultra-WideBand					
UWB	2007 [96, 97]	Automatic	200 m	27 Mb/s	3.1-10.6 GHz

* Some channels require *manual* human supervision to be used, whereas some others *automatically* discover peers in the vicinity.
** The range of a radio signal depends on its frequency (which drives signal attenuation) and on the antenna's power and type (directional or omnidirectional), not on its underlying standard [31]. The maximum distances presented here refer to typical user equipment.

vides a TV remote control application that leverages its smartphones' infrared sensors [230]. Infrared communication requires that the emitter and receiver be in direct line of sight. Depending on the source intensity, the maximum communication distance can span miles. Most of infrared's commercial applications employ proprietary protocols, although the *Infrared Data Association (IrDA)* has provided a standard protocol since 1994 [100], offering a data rate of up to 16 Mb/s.

[230] Xiaomi Inc. (2022), Mi Remote controller - for TV, STB, AC and more – Apps on Google Play

[100] Infrared Data Association (2001), Serial Infrared Physical Layer Specification

QRcode-flashing. Recent years have seen the advent of a one-way D2D communication channel—so ubiquitous it does not even have a name: shooting a Quick-Response (QR) code from a device's screen to another device through its camera. The practice arises from QR codes, a well-settled technology that has already found worldwide adoption. As a replacement for lower-capacity barcodes, QR codes are widely used for product tracing, loyalty programs, and similar applications. It enables augmented reality applications to display restaurant menus or locate objects in 3D space. Finally, in recent years, the QR code-flashing channel has seen widespread adoption, as it is pervasively used as an authentication technique for, e.g., travel tickets, Wi-Fi log-in, and, most recently, Covid-19 contact tracing and access control [34].

According to its ISO specification [98], a QR code is capable of holding up to 23.6 kb of data—depending on the code's size, data type, and level of *Error Correction Coding* (ECC). Specifically, as a means to transfer information from a screen-enabled device to a

[34] Campbell (2021), "QR Code Contact Tracing, Vaccine Passports Becoming New Normal in Europe, America"

camera-enabled one, it intuitively seems impractical to use the QR code-flashing D2D channel as a data-streaming interface. Indeed, generating, capturing, and interpreting a QR code is somewhat cumbersome and time-intensive. For the time being, let us only consider the channel as a means to exchange one-way *pulses* of data.

WI-FI

In 1996, Chai-Keong Toh deposited a patent for routing ad hoc mobile networks, published in 1999 [208]. In 1997, the IEEE 802.11 Working Group published its first standard for wireless network communication in the 2.4 GHz frequency band [88], inside which the 'ad hoc network' mode was a first-class citizen. Considered as 'beta', this initial standard was superseded in 1999 by 802.11a [89] and 802.11b [90], which were widely adopted by manufacturers and eventually led to 'Wi-Fi' becoming the ubiquitous wireless medium we know today.

[208] Toh (1999), "Routing Method for Ad-Hoc Mobile Networks"

'Ad hoc networking' enables direct communication between end-devices, that is D2D communication. Additionally, it supports multi-hop routing: two out-of-range devices can communicate via Wi-Fi ad hoc using an intermediary gateway device, effectively enabling *Mobile Ad hoc Networks (MANETs)* [42]—which is outside the scope of this chapter.

[42] Corson et al. (1999), Mobile Ad Hoc Networking (MANET)

Lastly, Wi-Fi ad hoc supports a mesh topology, where any two devices connected to a Wi-Fi ad hoc network can communicate directly with one another without going through a gateway node, as long as they are in range. It is notable that D2D communication was seriously considered by pioneers of Wi-Fi from its inception, and even beforehand. Wireless network card manufacturers implemented the functionality, leading to academic experiments on ad hoc networks from 2000 onwards, notably by the aforementioned patent holder, C.-K. Toh [206, 207]. Because 'Wi-Fi ad hoc' exists since Wi-Fi's inception, it features the same capabilities as the most recent Wi-Fi standards' implementations: as of 2021, the information rate climbs up to 20 Gb/s at close range using the 60 GHz band (standard 802.11ay [93]), while it maxes out at 9,608 Mb/s on the traditional 4.2–6 GHz band (standard 802.11ax, also called Wi-Fi 6E [92]). Concurrently with IEEE 802.11's standardisation efforts, the Wi-Fi Alliance periodically proposes standards that enable new applications and improve the Wi-Fi user experience.

[221] Wi-Fi Alliance (2023), Discover Wi-Fi Direct

Wi-Fi Direct. 'Wi-Fi Direct' [221] (formerly 'Wi-Fi P2P') was introduced in 2010 by the Wi-Fi Alliance, in an initiative to make ad hoc networks simpler. This specification aligns Wi-Fi operations more closely with Bluetooth for use cases like device pairing, while leveraging the superior bandwidth of the Wi-Fi family. Wi-Fi Direct restricts itself to single-hop communication in a star topology (all communications go through the Wi-Fi Direct gateway, coined the Group Owner).

By reusing Wi-Fi features from previous standards, only one Wi-Fi device needs to be compliant with Wi-Fi Direct.

Wi-Fi Aware. 'Wi-Fi Aware' [223] was introduced in 2015, mostly as a tool aggregating Wi-Fi and *Bluetooth Low Energy (BLE)* to deliver energy-efficient advertisement & discovery of services in proximity. Users can read insightful data from 'beaconing' devices (i.e., devices that emit only short pulses of information) without connecting to them, although the specification also allows switching to a connected mode in a mesh topology (no group owner). The Android kernel has supported Wi-Fi Aware since 2017 [73], suggesting that Google may have considered it as a beaconing back-end. However, no other OS supports Wi-Fi Aware, effectively making it an Android-specific artifact.

[223] Wi-Fi Alliance (2023), Wi-Fi Aware

[73] Google (2023), Wi-Fi Aware Overview | Android Documentation

BLUETOOTH

Bluetooth is another ubiquitous wireless communication channel utilising the unlicensed 2.4 GHz radio band. Following its first specification by the Bluetooth *Special Interest Group* (SIG) in 1999 [27], the first Bluetooth-enabled consumer device was a hands-free headset, highlighting Bluetooth's core objective: to obsolete cables for connecting digital peripherals—a typical D2D use case. At the time of writing, Bluetooth 6.2, published in November 2025, is the latest of 17 *core* specifications of the protocol [30, 32].

[27] Bluetooth SIG (1999), Bluetooth Core Specification

Bluetooth standards are split into tens of *profiles* [224], segregated per usage scenario. Profiles are not necessarily tied to a specific core specification. Each Bluetooth-enabled device only needs to support a subset of the available profiles—e.g., '*Advanced Audio Distribution Profile* (A2DP)' for a wireless headset—facilitating Bluetooth implementation for device manufacturers. Bluetooth creates point-to-point connections, called *pairs*, having a primary/secondary relationship. Bluetooth connections support up to 1–3 Mb/s date rates. A primary device can concurrently maintain connections with up to seven secondary nodes, resulting in a star-topology network coined a 'piconet' [188].

[224] Wikipedia (2022), List of Bluetooth Profiles

[188] Sairam et al. (2002), "Bluetooth in Wireless Communication"

Bluetooth Low Energy (BLE). BLE is a protocol stack that enables Bluetooth to operate on small, low-power devices (0.01–0.50 W). It enabled the cheap manufacture of 'beacons': small devices that broadcast short bursts of information (they never establish a full connection). This class of small-scale devices opened the door to new use cases, such as indoor geolocation (where BLE beacons are deployed to enable Bluetooth-enabled devices to locate themselves). BLE being a different protocol from 'classical Bluetooth', they are incompatible. However, most appliances supporting the latter (e.g., smartphones) are 'dual-mode': they also implement the BLE protocol.

Bluetooth and Wi-Fi may appear to be competing for the wireless consumer market: Wi-Fi supports much higher data rates, while Blue-

tooth's profiles make it cheaper and easier to implement on low-end IoT devices. However, several proposals leverage both channels (e.g., Wi-Fi Aware), suggesting they could be combined to improve overall D2D connectivity.

RFID

Operating in MHz bands at data rates in the order of kb/s, Radio-Frequency IDentification (RFID) is a particularly inexpensive short-range wireless channel. It has been commercially deployed for decades as a remote identification technology for, *e.g.*, theft prevention, goods inventories & access-control through badges [22]. This is thanks to RFID 'tags', which are able to emit beacons using only the RF energy harvested from their environment, without a battery [219, 220].

[22] Bays et al. (2017), Use of RFID for Tracking Government Property - Proof of Concept/Pilot

In 2004, the ISO standardised Near-Field Communication (NFC) [95], a novel communication protocol based on RFID. Most modern smartphones nowadays fully support NFC. Operating at 13.56 MHz with a maximum throughput of 424 kb/s, NFC allows contactless communication between devices over a few centimetres. This short-range requires little power and avoids interference issues. Due to its low throughput, NFC is a bad fit for demanding data exchanges. However, it is a very compelling channel for secure authentication, as evidenced by its broad adoption for contactless payments and the latest Wi-Fi authentication schemes.

UWB

Ultra-wideband (UWB) is characterised by its wide frequency band (500 MHz). It transmits several bits of information in a single *pulse* over a range of frequencies. This operating mode differentiates it from most other wireless channels, which exchange a bit per time step over a single frequency. Despite being one of the first radio communication channels, UWB was only granted an authorisation for commercial use by the FCC in 2002, on the unlicensed 3.1–10.6 GHz frequency band [41, 164]. The ISO has published UWB standards since 2007 [96, 97]. It has been sponsored by the UWB Alliance and the FiRa Consortium since 2019. IEEE's working group 802.15 on *Wireless Personal Area Networks* (WPANs) has been working towards a specification since 2020. Deployment-wise, the iPhone 11 (2019) was the first consumer mobile device to support UWB [14], thanks to Apple's U1 chip.

[14] Apple Support (2021), Ultra Wideband Availability

As a D2D communication channel, UWB is advertised as a low-energy, high-throughput (27 Mb/s) channel supporting a range of several hundred meters. It is also expected to become a precise geo-location and tracking technology [41].

DISCARDED CHANNELS

According to our definition of D2D communication, ZigBee [91], Li-Fi [103], LoRa [19], and SigFox [114] are outside the scope of this

chapter: they require dedicated hardware that is not widely available on consumer mobile devices. Some of these protocols additionally require a third-party coordinator to exchange data between two peers. Nevertheless, they are worth considering in broader D2D studies for several reasons: Li-Fi avoids radio interferences by working on the visible light spectrum; the *Low Power Wide Area Network* (LPWAN) properties of LoRa & SigFox are particularly interesting for the IoT sector [137] and geo-location [5]; and ZigBee—intended to be a simpler protocol than Bluetooth & Wi-Fi—can be a good fit on low-end embedded devices (e.g., sensors).

[137] Mekki et al. (2018), "Overview of Cellular LPWAN Technologies for IoT Deployment"

[5] Aernouts et al. (2018), "Sigfox and LoRaWAN Datasets for Fingerprint Localization in Large Urban and Rural Areas"

4.1.1.2 *In-band channels*

Cellular communication is regulated by the Global Standards Organisation 3GPP (for '3rd Generation Partnership Project'). The 3GPP began working on D2D communication in 2011 as part of Release 12 (Rel12) [1], which was frozen in 2015. The focus was put on public safety & critical communication sectors (as a replacement for the legacy Land Mobile Radio channel or LMR), resulting in two 3GPP 'Work Items': *Proximity Services* (ProSe), and Group Communication. The integration of emerging D2D use cases into subsequent 3GPP releases continues. The latest frozen release, Rel15 [2] notably took interest in *Internet of Things (IoT)*, *Machine Type Communication* (MTC) and *Vehicle-to-Everything* (V2x) use cases. Although 3GPP protocols, such as *LTE for Machines* (LTE-M) and *NarrowBand-IoT* (NB-IoT), support direct communication between devices (without going through a cellular base station) [105, 107], hardware support for this is entirely absent from end-user appliances.

4.1.1.3 *Proprietary solutions*

Specialised devices typically feature ad hoc wireless communication protocols, although these protocols are generally not available on mainstream end-user equipment. An iconic example is Apple, which is well-known for streamlining its users' experience across devices (e.g., iPhone, iMac, Apple TV…) through proprietary protocols.

Particularly interesting in the context of D2D communication are the AirPlay & AirDrop services, due to their wide adoption and acknowledged performance. AirPlay [16], dating back to 2004, is limited to streaming audiovisual content from computers (including phones and tablets) to TVs and speakers. AirDrop, on the other hand, is a general-purpose file transfer solution. It was initially released in 2011 and can only be used between Apple devices. Both services are backed by the undocumented, proprietary *Apple Wireless Direct Link (AWDL)* protocol, and leverage standard close-range wireless channels: Wi-Fi & Bluetooth, and UWB since iPhone 11 [41]. The *Open Wireless Link* (OWL) project [196] has been reverse-engineering AWDL since 2018 [195]. The

[16] Apple (2022), AirPlay

[41] Cooper (2020), Ultra Wideband & You

[195] Stute et al. (2018), "One Billion Apples' Secret Sauce"

OWL project's ongoing work includes the publication of security flaws in AWDL [197] and the open implementation of AirDrop, called OpenDrop [191].

[197] Stute et al. (2019), "A Billion Open Interfaces for Eve and Mallory"

Since 2018, Google has also proposed a proprietary D2D solution for Android phones, entitled *Google Nearby Connections (GNC)* [72]. As part of the Android Play Services, it is therefore installed in most Android phones. Contrary to AirDrop, GNC exposes APIs to application developers, allowing any Android app to establish general-purpose direct connections between phones (and even groups of phones). Antonioli *et al.* [11] reverse-engineered GNC in 2019, exposing severe security breaches in the same paper. The paper shows that GNC operates similarly to AWDL.

[72] Google (2022), Nearby Connections API

[11] Antonioli et al. (2019), "Nearby Threats"

Because they are undocumented and only use standard channels, AirDrop and GNC were excluded from table 4.1. However, these two proprietary channels demonstrate that performant and reliable D2D communication is feasible and largely requires the relevant OS support.

4.1.1.4 *Hardware*

GENERAL CASE

Since the iPhone 1 [13] in 2007, smartphones have always supported at least cellular, Wi-Fi, and Bluetooth communication. At the time, mobile phones used to contain one chip per channel. Nowadays, however, most mobile phones contain a single circuit chip (*i.e.* a *System on a Chip (SoC)*) that integrates most of the features needed for a smartphone—i.e., CPU, GPU, volatile memory, network connectivity, camera...

One of the most widely deployed SoCs as of 2026 is the Snapdragon series from Qualcomm, which supports Bluetooth, Wi-Fi, cellular, and NFC communication. Another one is MediaTek's Dimensity series, which offers the same connectivity support. Since 2021, Google has designed its own SoC, called Tensor, which is shipped in the Google Pixel series of smartphones from version 6 onwards [156]. Based on a system parameter to disable UWB on the Pixel 6 Pro [115], the Tensor SoC supports UWB in addition to the above-mentioned channels. Lastly, some SoCs with no connectivity support also exist, such as HiSilicon's Kirin. Smartphones equipped with the latter host additional hardware for wireless communication.

[156] Nieva (2021), Google's Pixel 6 Phones Are Coming with a Chip Designed in-House

In the end, all smartphones support Wi-Fi and Bluetooth, the most ubiquitous out-of-band channels for D2D communication. Hardware support for UWB is still nascent. None of the current smartphone SoCs support in-band channels, such as LTE-M or NB-IoT, nor atypical channels like ZigBee.

THE SPECIAL CASE OF APPLE

Apple devices are now fully designed by Apple: from hardware to

OS to system APIs. Nevertheless, Apple's developments are generally closed-source, making it hard for the most curious among us to scrutinise the operations of its products. As previously presented, Apple's AirPlay and AirDrop D2D protocols are backed by the Apple Wireless Direct Link (AWDL) D2D protocol, which functions atop standard Wi-Fi chips according to the OWL project [195, 196]. Since 2019, iPhones are packaged with the U1 network chip [17], which additionally supports the UWB channel [21].

[21] Barrett (2019), "The Biggest iPhone News Is a Tiny New Chip Inside It"

4.1.2 *Security models*

Compared to the Internet, the unique properties of D2D communication impose specific security challenges. On one hand, D2D locality obviates most remote attack scenarios, which are daunting in the Internet paradigm. On the other hand, the broadcast nature of D2D wireless channels and the need for periodic self-advertisement introduce new attack classes. This Section provides an overview of the security model for D2D communication. Several surveys propose a more in-depth study [83, 218].

This Subsection introduces D2D attack scenarios and then enumerates the desired security properties. Then, it presents D2D-specific security features. Finally, as an example of these features, this Subsection briefly describes the current authentication schemes for Bluetooth and Wi-Fi.

4.1.2.1 *Attack scenarios*

PASSIVE ATTACKS. Due to the broadcast nature of wireless communication, an adversary *in the vicinity of a user* may passively eavesdrop and/or analyse D2D traffic to learn sensitive information about other peers.

Unencrypted messages are directly accessible through *eavesdropping*, while *traffic analysis* can infer context information, even on encrypted traffic, such as the object of a transmission (text, file, stream…) or the service being used. Location information can also be passively inferred, such as D2D users' whereabouts and travel patterns [168].

[168] Primault et al. (2019), "The Long Road to Computational Location Privacy"

ACTIVE ATTACKS. An active adversary may perform a *Denial of Service* (DoS) attack by *jamming* the communication medium. By *forging* and/or *manipulating* D2D messages, a malicious user within transmission range can further trick honest peers: they may access unauthorised services, impersonate other persons, or even steal information from their victims.

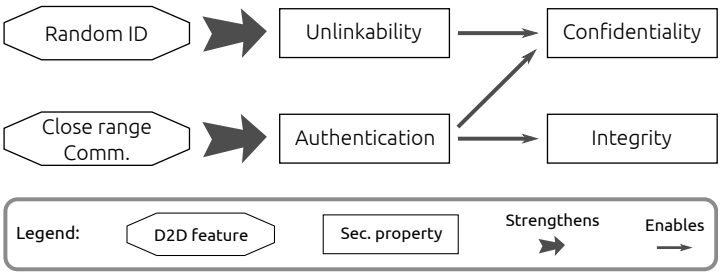


Figure 4.1: Some of D2D communication's features strengthen our desired security properties.

4.1.2.2 *Desired security & privacy properties*

D2D security as a whole is driven by network security & privacy requirements. The two concepts are closely related, though they sometimes conflict.

NETWORK SECURITY. The three usual tenets of network security are: **availability**, the fact of ensuring that the network is always accessible; **integrity**, which protects users against the modification or falsification of exchanged information; and **confidentiality**, ensuring that only authorized users can read a piece of information [192]. This latter is enabled by *cryptographic authentication*, which provides for both *identification* and *encryption* [193]. Encryption is of paramount importance in wireless networks, as any eavesdropper within range can read unencrypted traffic.

[192] Shen et al. (2014), "Secure Key Establishment for Device-to-Device Communications"

[193] Shiu et al. (2011), "Physical Layer Security in Wireless Networks"

PRIVACY. Privacy does not have a widely accepted definition, but it can be understood as a personal decision about which personal information one agrees to share with whom, how, and to what extent. There is an inherent trade-off between privacy and utility, as the utility of a service generally increases as it is provided with more personal information on its user [83]. In D2D communication, the major pillars of network privacy are: **user anonymity**, or hiding from unauthorised peers any information that could be used to find one's identity; **context privacy**, which prevents attackers from learning context information about a communication (e.g., location, service being used, query type); and **unlinkability**, which aims at making several communication sessions by the same user indistinguishable from sessions by different users (one's communication history cannot be *linked*). As opposed to user anonymity—that takes interest in the messages' *content*, unlinkability takes interest in the *way* messages are sent [119].

[83] Haus et al. (2017), "Security and Privacy in Device-to-Device (D2D) Communication"

[119] Lindell (2011), "Anonymous Authentication"

4.1.2.3 *D2D-specific security features*

Figure 4.1 depicts how D2D-specific features enable the aforementioned security properties. We focus on network security properties, and not privacy ones, as the former have more consensual definitions.

RANDOMIZED HOST IDENTIFIERS (RANDOM ID). Internet communication stands atop the IP protocol of the third 'network' OSI layer. In the IP protocol, the IP address behaves like one's postal address on the net. It is a mandatory piece of metadata to include when communicating; without it, your addressee cannot respond. However, this piece of metadata jeopardizes *unlinkability*, as it identifies a user and can thus be easily used to track individuals online [141]. Hiding one's IP address remains possible through anonymity networks like Tor [49], but it is a cumbersome process, mostly undertaken by people in need of serious anonymity (e.g., journalists and whistleblowers).

[141] Mishra et al. (2020), "Don't Count Me Out: On the Relevance of IP Address In The Tracking Ecosystem"

[49] Dingledine et al. (2004), Tor

On the other hand, we defined D2D communication as 'single hop' communication, taking no interest in routing messages through networks: D2D communication occurs at the first ('physical') and second ('data link') network layers. Most often, data link is achieved through the Medium Access Control (MAC) protocol. Similarly to IP, a layer-2 address is required to identify communicating hosts, notably in the advertising (broadcast) phase. Privacy-wise, this situation is bad because such an address identifies a network card— and, consequently, its host device and, by extension, its user. However, *there is no crucial need to broadcast one's* real *layer-2 address to establish and maintain a D2D connection.* Instead, this address can be randomly generated and periodically rotated by their host device to strengthen *unlinkability*. Efforts are indeed undertaken to hide this personal identifier by the most famous D2D channels:

- In the Wi-Fi realm, the effort to randomise the MAC address has initially been taken up by OS manufacturers, such as Linux Tails [201]. Although IEEE 802.11 has been pushing MAC randomisation since 2018 (as part of amendment 802.11aq), and created a task group entitled 'Randomised and Changing MAC Addresses' in 2021, specifically interested in anonymising MAC addresses without loss of utility for related services [94]. Indeed, several network operations traditionally depend on fixed MAC addresses, such as MAC filtering and DHCP reservations.

[201] Tails (2022), MAC Address Anonymization

[94] IEEE (2022), IEEE P802.11 - TASK GROUP BH (RCM)

- The Bluetooth 4.0 specification introduced the 'smart privacy' feature, which notably replaces one's broadcast MAC address with a random one derived from the real Bluetooth card MAC address [28]. The only way to retrieve the real MAC from the random one is using the broadcaster's *Identity Resolution Key (IRK)*. The IRK is shared with remotes only after pairing, meaning only trusted peers can de-anonymise one's broadcast MAC address.

[28] Bluetooth SIG (2015), Bluetooth Technology Protecting Your Privacy

Unlinkable communication sessions benefit *confidentiality* by hiding real network identifiers from unauthorised parties. Furthermore, such randomised & evolving addresses preclude longitudinal location tracking, and thus also improve *context privacy*.

CLOSE RANGE COMMUNICATION. Digital authentication is the process of verifying one's identity to determine whether they are authorized to access specific information. Authentication revolves around the exchange and validation of cryptographic keys, which are then used to encrypt and sign future messages. In that regard, authentication enables both the *confidentiality* and *integrity* security properties.

Three factors are generally used to establish one's digital identity: **knowledge**, where the system verifies that the user *knows* a private information (password, PIN codes...); **possession**, where the system verifies that the user *owns* a physical object (computing device, token...); and lastly the **biometric factor**, where the system verifies something that the user *is* (fingerprints, face, eye scan...). We use Wi-Fi or Bluetooth to exchange passwords, which constitute *knowledge factors*. Yet, authentication can be further facilitated by using physical proximity to assert a *possession factor*. *Close-range communication* (e.g., NFC or QR code scanning) is a convincing way to ensure that communication is initiated between two legitimate users, as both assert that they are the ones *owning* the communicating devices. Furthermore, close-range communication can be viewed as a bootstrap side-channel for exchanging encryption keys, which removes the need for e.g., Diffie-Hellman key exchanges (prone to man-in-the-middle attacks) and establishes a second authentication factor: *knowledge*.

Note that authentication does not necessarily imply the disclosure of one's *full* identity. Through cryptography and with the help of a trusted source (such as a certificate authority), it becomes possible to ensure that one is authorised to access a service without disclosing any other information. The process is called *anonymous authentication* [56, 119], but cannot be easily carried out without the help of a third-party (outside the D2D communication scope).

4.1.2.4 *Authentication in Bluetooth and Wi-Fi*

Since their inception, both channels have supported open or password-protected peer authentication. While Wi-Fi networks use long-term passwords, Bluetooth sometimes uses one-time PIN codes (which only need to be matched on both pairing devices). In Bluetooth, the initial authentication (pairing) is usually manual. Because devices exchange encryption keys (such as the aforementioned IRK) during pairing, connecting already paired devices is automatic and much simpler: the devices recognise and connect to each other without human intervention. In 2007, the Wi-Fi Alliance proposed *Wi-Fi Protected*

Setup (WPS) to ease network onboarding, notably featuring a PIN code scheme similar to Bluetooth's. Sadly, WPS' PIN code mode contained a vulnerability that allowed an attacker to brute-force the PIN code under certain conditions [213].

Wi-Fi Easy Connect (WEC) [222] was introduced in 2018 to further ease network authentication of interface-less IoT devices. WEC connects such a device (called 'enrollee') to a wireless network by using a third-party 'configurator' (smartphone or otherwise). By scanning a QR code or an NFC tag on the enrollee, the configurator initiates a secure connection to exchange information about the Wi-Fi access point and the connecting device. Through the configurator, the enrollee receives clearance and can finally connect to the network independently. This process constitutes an example of close-range communication for authentication, as discussed above.

[213] Viehböck (2011), "Brute Forcing Wi-Fi Protected Setup"

[222] Wi-Fi Alliance (2023), Discover Wi-Fi Easy Connect

4.1.3 *Software Hurdles*

4.1.3.1 *Increased development costs*

There are several ways to implement D2D communication, such as Bluetooth or Wi-Fi, but **there is no unified middleware** that provides high-level programmers with access to different D2D schemes. Vendors need to further develop their offerings to enable their efficient use. We believe that their lack hinders developers and novel D2D applications.

LACK OF NETWORKING STACK. The absence of a unified middleware requires implementing all the low-level network routines; for instance, a file-sharing application relying on D2D must be implemented from scratch on the corresponding OS. This involves low-level details, such as choosing the most appropriate communication channel. We should not expect front-end and app developers to satisfactorily address these issues, which are outside their fields of expertise. The network stack should be developed by networking experts and integrated by vendors.

Several specialized D2D communication protocols are already widely deployed, such as Chromecast or Miracast for audio/video streaming. However, such solutions are often proprietary (i.e., not interoperable), and lack the generality needed to enable *any* type of D2D data dissemination. Their specialization, however, makes them appealing to manufacturers that only need to implement these relatively simple protocols, rather than, for example, the entire Wi-Fi stack.

LACK OF TESTING TOOLS. The absence of a unified middleware results in a lack of supporting tools, particularly for testing. Connectivity affects *Quality of Service* (QoS). Services and applications must be properly tested, in particular when connectivity is faulty. A service

should respond despite intermittent or slow connections, interference, and other issues that arise when using D2D communication. It is essential to rely on a testing platform to automatically assess QoS under different scenarios.

Despite many methods for simulating the impact of Internet connectivity on QoS, there are no such tools for D2D communication. The XCode IDE provides tooling to test slow Internet connections but does not support D2D connections, such as Bluetooth. Similarly, R2Lab [160] is a wireless test bed for studying Wi-Fi and 4G/5G networks in anechoic chambers, but it does not support other D2D technologies. While some academic frameworks exist to test nearby *Peer-to-Peer* (P2P) mobile apps [136], they assume stable D2D connections once devices are close enough.

[160] Pakzad et al. (2018), "R2Lab Testbed Evaluation for Wireless Mesh Network Experiments"

[136] Meftah et al. (2019), "Testing Nearby Peer-to-Peer Mobile Apps at Large"

Developing a generic D2D testing framework is hard because it must work across many configurations, including platforms (e.g., Windows, iOS, Android, Linux) and channels. However, such a framework is an essential prerequisite to support developers in building reliable and trusted D2D services.

4.1.3.2 *No incentive to decentralize*

Despite reducing infrastructure costs, it is unclear whether the effort to support D2D communication increases profits.

MORE CONTROL MEANS MORE REVENUES. Centralized services have many advantages. It can be used to force users to update the software, to control access to the service, and to collect and monetize personal data. In the era of big data, designing a non-centralized service makes it harder to monetize.

DEVELOPER BIAS: THE INTERNET IS RELIABLE, FAST, AND CHEAP. From a developer's point of view, Internet access is usually reliable and fast, even for mobile devices. Moreover, the user pays the fees. Deploying Internet services thus seems simple and cheap: it mostly requires a server with Internet access. Since a server is required anyway, the potential gain from adding D2D communication might be offset by increased development costs.

This bias burdens users who pay for their Internet traffic by volume or have unreliable connections. For them, D2D communication would not only be more efficient, it would also be cheaper. For example, exchanging large files over the Internet can be expensive or impossible, depending on fees and connectivity, whereas it would be free and convenient between nearby devices via D2D communication.

4.1.3.3 *Operating systems are not interested*

Operating systems (OSes) play a central role in networking by providing the implementations of the APIs and protocols used by network cards. Application developers depend on support from the OS: *(i) when the OS does not support a communication channel*, applications cannot use it; *(ii) when the OS only provides low-level communication APIs*, each application must ensure reliable and efficient communication by itself; and *(iii) when the OS provides high-level D2D abstractions*, which ensure robust, performant, and energy-efficient communication, all applications can benefit from the feature. **The policy chosen by the OS vendor affects what the developers can do.**

D2D COMMUNICATION IS NOT A PRIORITY FOR OSES. It might not seem worthwhile for an OS to support D2D communication channels—as shown by the poor user experience of sending files over Bluetooth (see Section 4.1.4). This makes developers reluctant to rely on them, which justifies vendors' decision not to improve their barely used D2D APIs, creating a cycle.

However, Apple's AirPlay & AirDrop protocols prove that D2D communication can be an asset. The company has a commercial advantage: all of its devices can exchange files, configurations, and even credentials directly and efficiently. Google released a similar Nearby Share protocol in 2020 [64], nearly nine years after the initial release of AirDrop, but it is not interoperable with the latter. There is no technical reason preventing D2D interoperability. This non-interoperability of the protocols is likely used as an additional way for vendors to tie their user base to their unified devices and platforms: MacOS and iOS *via* AirDrop, and Android and Chrome *via* Google Nearby Share. However, for D2D communication to reach mass adoption, OSes must provide high-level, cross-platform D2D APIs.

[64] Google Help (2022), Share Files or Apps with Nearby Share

4.1.3.4 *Connectivity challenges*

The main challenge in exchanging data via D2D communication is establishing the channel. The key issues are device discovery and synchronization in the absence of a centralized coordinator (in cellular networks, this role is played by infrastructure equipment, such as base stations). Without the coordinator, more complex algorithms are required to ensure discovery and synchronization. Even without this infrastructure, distributed synchronization can be achieved in different ways [4, 54, 199].

Device discovery is the process of finding nearby devices to communicate with. Without centralized infrastructure, distributed device discovery can be a challenge. Similarly to clock synchronization, it is generally performed by devices periodically broadcasting messages to announce their presence [84, 104]. The receipt of such messages helps

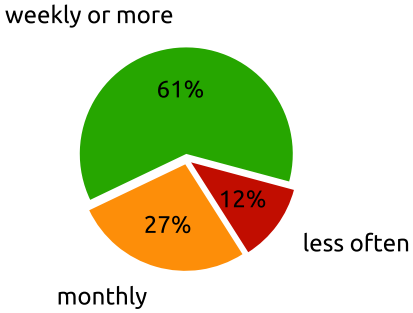


Figure 4.2: Distribution of the frequency of the need for D2D communication. 61% of the surveyed users need D2D communication at least once a week, while only 12% need it less than once a month.

build a local view of the network. Interference may occur when devices use the same frequency [132], which is likely to happen because most D2D communication channels operate within limited frequency ranges. In particular, many collisions occur when using the 2.4 GHz band, which is widely used by Wi-Fi and Bluetooth. Dealing with interferences is an open problem, especially in energy-constrained mobile environments, but practical solutions already exist [24, 153, 235].

[132] Martins et al. (2021), "Interference Cancellation Techniques for Device-to-Device Discovery in Out-of-Coverage Networks"

4.1.4 *The discouraging D2D user experience*

We conducted an online survey on 'how people usually share documents between physically close devices', available at `https://d2dsurvey.luxeylab.net`. along with the raw survey data. This use case could typically be undertaken using D2D protocols. Our goal was to estimate how the public achieves it and their satisfaction with the available solutions. After a brief inquiry about their IT know-how, participants were asked how often they need to exchange documents between nearby devices. Then, they were asked to express how often they use each and their level of satisfaction with Bluetooth, external storage (ext), cloud services, and, for Apple users only, AirDrop. A free-text field allowed participants to provide comments and feedback. We gathered 364 answers, including 64% professionals of the IT community.

THERE IS A NEED FOR D2D COMMUNICATION. Figure 4.2 shows the distribution of the frequency of the need to share files between nearby devices. 61% of the participants face the use case weekly or more, while only 12% need D2D file sharing less than once a month. This figure highlights the ubiquity of media exchange in our digital lifestyles: from lolcats and memes, to text documents (administrative, press...), including voice recordings, souvenir pictures and videos, etc.

Table 4.2: Alternate solutions used by participants of our survey for close file sharing, proposed in the free-text field (66 participants left an answer). Technologies are not exclusive, e.g., *Samba* is generally a *Windows Local net* solution.

Solution name	**Count of occurrences ↓**
USB	16
Local net	13
Syncthing	9
CLI	8
Windows	4
KDE Connect	3
Samba	3
NAS	2
sharedrop.io	2
Warpinator	2
beam.cafe	1
DLNA	1
NitroShare	1
snapdrop.net	1

MANY ALTERNATE SOLUTIONS TO A SIMPLE PROBLEM. Out of the 364 participants, 66 left an answer in the free-text feedback field. As shown in table 4.2, people address the close-file-sharing issue within their entourage using many alternative solutions: D2D or not, ad hoc or purposeful. Notably: Syncthing,[1] Samba, `sharedrop.io`, KDE Connect, Warpinator, proximity sharing with a Wi-Fi Hotspot (Local net), `rsync` (CLI), etc. This variety can be interpreted in different ways. On the one hand, it shows the versatility of the digital tool, as it enables many different ways to solve a simple task. On the other hand, it also highlights the lack of a universal D2D solution for file exchange.

EXISTING D2D TECHNOLOGIES ARE NOT SATISFACTORY. Figure 4.3 reports on the percentage of participants who frequently use each technology and their level of satisfaction. Except for AirDrop, satisfaction percentages are computed based on the total number of participants, not only on those who frequently use the given technology, since dissatisfaction with a technology affects its usage frequency. For AirDrop, the percentages are computed only for Apple users.

Only 6% use Bluetooth frequently, and only 17% are satisfied with it. While AirDrop seems to be a better D2D alternative for Mac users, 42% of Mac users use it frequently and 60% deem it satisfactory. Non-D2D technologies are significantly preferred. 34% of the participants frequently use external storage, with 67% satisfaction. The most fre-

1 Syncthing exists precisely to allow file exchange without involving third parties. It works over the Internet or in a D2D fashion if the involved devices are on the same local network. See `https://syncthing.net/`.

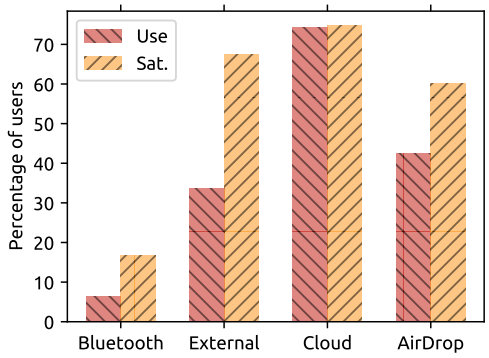


Figure 4.3: Use and satisfaction (Sat.) of the different available technologies. Only users who frequently use the given technology are considered. Bluetooth is the least popular technology for exchanging documents between nearby devices, despite being designed for this purpose. On the other hand, the cloud is the preferred solution, as it is the only one that does not rely on the device's proximity.

quently used technology for exchanging documents between nearby devices is, ironically, the cloud: 74% of the participants use it for that purpose, and 75% are satisfied with it.

A first observation is that, despite the need for D2D file sharing, people rarely use Bluetooth or Airdrop. They prefer other sharing means (external storage, or mostly the cloud). Airdrop has usage and satisfaction percentages of 42% and 60%, respectively, among Apple-using participants. Both statistics fall short of the cloud's. The lack of interoperability with other vendors may partly explain this underwhelming observation.

However, the real question lies in the minor 17% satisfaction towards Bluetooth, and its even lower 6% usage rate. This disuse is surprising, given that this channel is specifically designed for exchanging files. Regarding external storage, its 34% usage rate can be explained by the burden of using a storage proxy, including cables. However, participants are fairly satisfied with it: once plugged in, cable solutions are indeed reliable. Finally, regarding the cloud: Its 75% satisfaction rate is in line with the amount of investment and the many actors involved in network-based file exchange. However, following our paper's motivation, we deem that 74% *of the participants regularly resorting to the cloud for such a D2D scenario is bad news*.

For the sake of the environmental footprint of ICT, privacy, and users' data bills, professionals ought to work towards reducing this proportion in favour of D2D channels.

4.2 A COMMUNICATION FRAMEWORK TO REMOVE USAGE FRICTION

To promote the deployment of Device-to-device (D2D), ease its implementation by mobile developers and explore the software engineering challenges D2D presents, we propose a tool named Venice.

Venice is a device-to-device framework designed to allow communication between devices in the same vicinity.

One of the key features of Venice is its multi-platform architecture, which virtually allows communication between any kind of desktop and smartphone devices. Another advantage of Venice is its channel abstraction, allowing developers to directly program the interaction between agents, and not the technical implementation details, that vary from platform to platform and from one OS version to another.

This behaviour allows the framework to think abstractly of the available communication technologies, and serves our end goal of abstracting physical channels from developers and users, automating their selection so that a functioning D2D link can be spawned.

4.2.1 *Connection process.*

We take an interest in a local wireless communication involving two parties: an *advertiser* and a *discoverer*. The former offers a given service, while the latter searches for services. The advertiser is not necessarily the one who sends the data: the *sender* role can be assumed by either the advertiser or the discoverer. For clarity's sake, in this paper, we consider *an advertiser who wants to send data to a discoverer*.

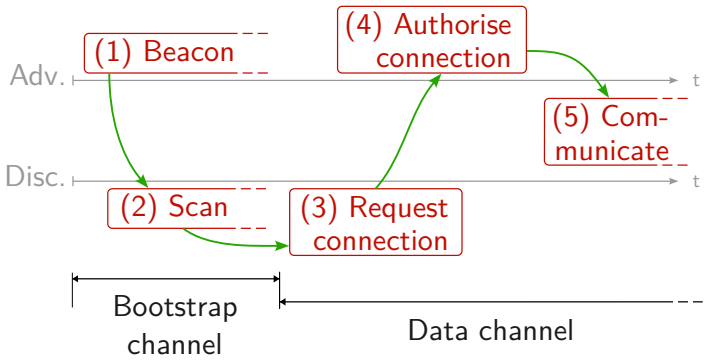


Figure 4.4: Device-to-device communication process. The *advertiser* (Adv.) offers a service, while the *discoverer* (Disc.) is looking for interactions. The service discovery is performed on a *bootstrap* channel, while the communication is performed on a *data* channel, see Table 4.3.

We describe the typical D2D connection procedure in Figure 4.4: (1) the advertiser locally advertises its availability for realising a task (*e.g.*, transmitting a photo) on a bootstrap channel; (2) the discoverer

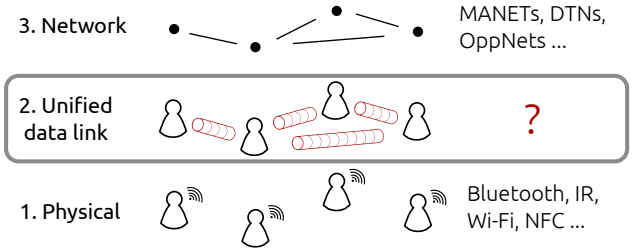


Figure 4.5: The literature lacks a standard abstraction to unify D2D channels in the link layer.

locally scans the same channel for available tasks; (3) the discoverer finds the advertiser's task, and would like to participate in it: they send a join request on the data channel; (4) the advertiser receives the discoverer's request, and decides to accept it; (5) communication is established between the two users: the task can start.

Some physical channels are better suited for discovery or connection, as shown in Table 4.3. Bluetooth, for instance, has a very low throughput in connected mode compared to Wi-Fi, but is widely used for connectionless advertising and scanning. In the wild, this bootstrap process is notably deployed in WEC [8] (that leverages a QR code to make a device join a Wi-Fi network through an already connected smartphone), and in the aforementioned GNC and AirDrop (that claim to transfer credentials on BLE to establish a Wi-Fi Direct connection).

[8] Alliance (2023), Wi-Fi Easy Connect

4.2.2 *A device-to-device communication framework*

As shown in Figure 4.5, D2D communication requires a unified layer 2 (link) to locate, connect, and transmit data among peers, regardless of the underlying physical channel (Bluetooth, Wi-Fi, etc.). Proposals towards that end do already exist (although not intercompatible), such as Apple's Airdrop/AirPlay, Google's Nearby protocols [72] or the Omni middleware [106].

[106] Kalbarczyk et al. (2018), "Omni"

VENICE aims to abstract the physical layer of local wireless communication, allowing developers to reason in terms of connection without having to delve into the specifics of the users' hardware stack. Being the missing piece between physical and network layers in a D2D setting, VENICE intends to play the role of the **OSI model "data link" layer 2 for D2D**.

Implemented using Google's cross-platform framework Flutter [69], VENICE presents a set of abstractions, including tech-agnostic schedulers; its core feature is the *channel* abstraction.

VENICE encapsulates each communication technology in a `BootstrapChannel` or a `DataChannel`, depending on whether it is used for connection initialization or to transmit payload data. VENICE thus remains agnostic to concrete channel implementations, which can be implemented inde-

Table 4.3: Preferred communication type per channel

	Physical channel				
Chan. type	**QR**	**Infrared**	**RFID**	**Bluetooth**	**Wi-Fi**
Bootstrap	☑	☑	☑	☑	⊠
Data	☐	⊠ [100]	⊠ [128]	☑	☑

☐: not implemented; ⊠: possible; ☑: widely deployed.

pendently and used interchangeably. Channels can be used to transmit any kind of payload from the sender to the receiver; the application is thus responsible for implementing its data transmission scheduling algorithm once the D2D link is established.

VENICE was implemented in a demonstration app requiring manual selection of bootstrap and data channels, the same on both participating devices. Since VENICE uses a stream of bytes to transmit information between agents, any use case can be implemented with it; this application, however, focuses on file exchange.

4.3 DISCUSSION

4.3.1 *Threats to validity*

Two main arguments disprove the interest in D2D communication.

First, the electrical consumption of the beacon/scan process is pointed out. Scanning one's surroundings enables proximity sensing, and is thus paramount for D2D scenarios, although it consumes energy [51, 136]. When the activity is carried out once (e.g., to connect to a hotspot), such a power draw is acceptable. Some D2D protocols would nevertheless require a continuous background scan to function. This energy consumption can, however, be optimised, e.g., through careful selection of operational parameters [121, 136].

The second argument held against D2D communication is its *a priori* shallow economic prospects. We indeed cited direct communication's increased privacy as one of its features: as a result, less behavioural information can be harvested, thereby preventing data-driven economic models from building on it. Given the increasing stress on our centralised network infrastructure, leaving the door open to D2D developments would remain relevant. Instead, Application Programming Interface (API) support across the major OSes remains insufficient and even diminishing, as we will see now.

A major hurdle to the development of D2D is the embryonic state of the OSes' implementation of their low-level APIs, as well as the lack of standard (intercompatible) high-level abstractions to ease their operation. An ongoing investigation within our team shows that general-purpose D2D APIs are lacking on most OSes: either in coverage, in support/documentation, or because they are deprecated due to conflicting objectives. For instance, the only sources of information about Bluetooth file transfer or Wi-Fi Direct support on iOS are online forums [139, 190], which boil down to: the feature is not documented. On Android, trivial network inspection (e.g., fetching the IP address of one's Wi-Fi gateway) is hindered by the newer location schemes; furthermore, programmatically creating Wi-Fi hotspots now generates random SSID & password, which complicates development. We regret that the lack of economic incentives towards D2D communication not only discourages but also undermines their support for such widespread systems.

Regarding the framework implementation, we built VENICE using the Flutter SDK [69], Google's cross-platform app development framework. Using a single code base, Flutter natively compiles apps for Android, iOS, the web, and desktop platforms, and thus aligns with VENICE's goal of being a powerful universal abstraction layer. As such, our library is ready to use, and all of its components are publicly available in a dedicated open-source GitHub organization [178]. We

[178] Raes et al. (2023), Venice source code repository

succeeded in creating VENICE through trial and error: even with a cross-platform framework like Flutter, major differences between platforms and even OS versions become apparent during development. In fact, within Android itself, services and permissions change depending on the SDK version used: for instance, to use Bluetooth with devices running Android 11 (up to SDK 30) or older, the `BLUETOOTH` permission must be used; while, starting from Android 12, this permission is split into `BLUETOOTH_ADVERTISE`, `BLUETOOTH_CONNECT` and `BLUETOOTH_SCAN`, making the integration of Bluetooth as a communication means particularly tedious [63].

[63] Gómez et al. (2015), "A Recommender System of Buggy App Checkers for App Store Moderators"

4.3.2 *Future works*

We implemented all necessary features to demonstrate working applications, as detailed in the next Section. We plan to extend VENICE's capabilities by (i) increasing the supported platforms and channels; (ii) including a negotiation phase in the connection protocol, to allow devices to agree on a subset of technologies to use based on their own capabilities; and (iii) exploring whether transfer time can be improved by leveraging several data channels at once.

For the time being, VENICE requires manual selection of a single bootstrap and data channel, the same on both participating devices. Our end goal is to abstract physical channels from developers and users; their selection will be automatically negotiated in a later version, enabling a functioning D2D link to be spawned. For data channel aggregation, we intend to leverage control theory to increase network throughput by maximising the number of concurrent channels.

Once several data channels are implemented, control-theoretic techniques can be leveraged to boost network throughput by maximising the number of concurrent channels. VENICE should also be resilient to channel roaming and handle out-of-range connection losses.

Considered use cases include sporadic file-system synchronisation and using a device as a bridge to connect to services.

Additionally, as a high-level solution, the VENICE proof of concept is useful to test the interchangeability of communication channels, but is obviously a dead-end for mobile app businesses that cannot afford to develop what should essentially be a kernel module: the unification of D2D operations under a single API. This interface should autonomously select appropriate radio channels based on the applications' requirements and the heterogeneous communication capabilities of the corresponding devices.

Going a step further, a unified layer 3 (network) would also be beneficial for transparent multi-hop D2D routing [106]. It would leverage the actual implementation of two decades of research on infrastructureless networks, such as *Mobile Ad hoc Networks (MANETs)*, *Delay-Tolerant Networks (DTNs)*, and *Opportunistic Networks (OppNets)* [35, 42, 209].

4.3.3 *Conclusion*

This chapter examined the adoption status of D2D communication among end-users. We highlighted the importance of this communication paradigm for environmental sustainability, privacy, and disparities in access to broadband Internet. An online poll showed that usage scenarios where D2D communication would be beneficial exist in the wild. The state of practice in communication channels was then reviewed, along with the paradigm's security models. The path seems clear: device-to-device is needed, and many timely solutions exist with interesting security features. So why such a low adoption rate?

A major technical hurdle remains—D2D communication's underwhelming support in operating systems. All existing APIs are either channel-specific and low-level or proprietary. Apple has never allowed Bluetooth file transfer. Android's Wi-Fi ad hoc APIs are impractical (e.g., randomised hotspot SSIDs and passwords). The variety of Bluetooth's hardware & software implementations is often pointed to explain its underwhelming performance on certain device pairs. Further study is needed to quantify the extent of these alleged compatibility issues.

As illustrated in Figure 4.5, we believe that a higher-level cross-platform API encompassing several communication channels would ease development. Similar to the IP stack, such an API would abstract the physical layer below more practical link-oriented controls. Indeed, all D2D channels follow the same communication process: discovery (beaconing & scanning), optionally followed by an authorisation and a connected phase.

As a first step, we propose VENICE, a device-to-device framework enabling a variety of available radio APIs to allow communication between devices in the same vicinity; we subsequently underline how VENICE is only a first step towards local communication in a world where its implementation, both technical and in usage, is highly demanding.

Going a step further, a unified network layer would be beneficial for transparent multi-hop API routing. Two decades of research on infrastructure-less networks have scrutinised the topic, e.g., *Mobile Ad hoc Networks (MANETs)*, *Delay-Tolerant Networks (DTNs)*, and *Opportunistic Networks (OppNets)* [35, 42, 209].

We hope that our D2D contributions motivate the adoption of more local communication processes and provide a solid foundation for continuing to advance toward this end.

5

ENERGY USAGE

Previous chapters have introduced ways to repatriate data processing from the cloud to smartphones, to protect users' privacy. However, there is no free lunch: as suggested by Google [162], doing so could increase overall energy consumption.

Exploring smartphone energy consumption, Section 5.1 delves into the energy use of mobile software, allowing developers to compare Third Party Libraries (TPLs) to select the best one for their apps; Section 5.2 subsequently proposes a power model that leverages embedded energy probes to explore the energy drain repartition between smartphone components.

Contents

5.1 Measuring and comparing the impact of TPLs on app quality **90**
5.1.1 Methodology 90
5.1.2 Validation 95
5.2 Understand mobile hardware consumption repartition **102**
5.3 Discussion **106**
5.3.1 On the usage of EQuAT 106
5.3.2 Threats to Validity 108
5.3.3 Future works 109
5.3.4 Conclusion 110

5.1 MEASURING AND COMPARING THE IMPACT OF TPLS ON APP QUALITY

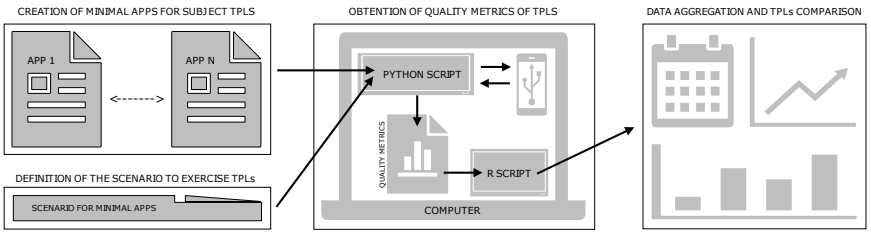


Figure 5.1: Evaluating Quality Attributes of TPLs (EQuAT) for comparing their impact on app quality.

[186] Saborido et al. (2018), "Getting the most from map data structures in Android"

Previous studies comparing Android data structures [186] and the impact of anti-patterns on mobile apps' energy consumption [149] have identified CPU, memory, and energy as relevant metrics for assessing the efficiency of Android implementations. These metrics are critical because they directly correlate with the overall smoothness of the user interface and the device's battery life.

Additionally, network activity is a decisive factor in distinguishing the resource overhead of different app versions and is often the primary driver of unexpected energy drain and data plan consumption [185].

[185] Saborido et al. (2017), "Comprehension of Ads-supported and Paid Android Applications: Are They Different?"

Thus, while CPU, memory, and network usages and energy consumption represent the operational cost, app size and permissions represent the architectural and privacy costs of TPLs. We consider these metrics as relevant quality attributes of mobile apps.

To measure and compare the impact of different TPLs on the quality attributes of apps we propose EQuAT, a four-step approach that automates running apps and collecting quality metrics on them.

5.1.1 *Methodology*

First, developers create a minimal app for each TPL under test (TPLUT). A minimal app is an app with the simplest Graphical User Interface (GUI) that includes the required components to exercise all the features of interest of the TPLUT. Second, they define as many scenarios as necessary to simulate a user's interaction with the TPLUT 's features. Then, developers apply EQuAT on the resulting file of each minimal app and the defined scenario to automatically obtain the size of each app, the number and type of permissions required for each app, and the performance (CPU, memory, network usage, and energy consumption) of each app over several runs. Finally, it aggregates the collected data, compares the TPLs using quality metrics, and ranks them accordingly. Fig. 5.1 summarises the proposed approach.

EQuAT takes as input a path to a folder containing the app files for minimal apps, and the scenarios to execute, each corresponding to a single app. It automatically runs the apps, plays the scenarios, and collects the quality metrics. It generates a comma-separated values (CSV) file containing all quality metrics for the TPLUT. Using this data, it aggregates quality metrics and generates plots to facilitate comparison of the TPLs. Each minimal app integrates one, and only one TPLUT, and all the minimal apps share the same configuration and GUI. Therefore, our approach can attribute any difference in metric values between any pair of minimal apps to the TPLs used by the apps.

In the following, we provide further details of the whole process for measuring and comparing the impact of TPLs on app quality. We focus on Android apps without loss of generality for the proposed approach.

CREATING MINIMAL APPS FOR TPLUT. EQuAT requires a minimal application for each TPLUT to ensure a fair comparison between different TPLs. Consequently, to spare developers the need to create minimal apps from scratch, we propose using category-specific app templates. Each template is derived from a blank Android project to exclude unnecessary permissions or GUI overhead.

Particularly, a template consists of *(i)* a configuration layer defining the minimum SDK version and the essential permissions required for that category (e.g., INTERNET for Ads or Analytics), *(ii)* a GUI baseline, and *(iii)* the initialization of the TPL to trigger its primary functionality. Thus, these templates act as a controlled baseline, ensuring that any measured differences in performance or resource usage are strictly attributable to the TPL itself rather than variations in the host app's structure.

Developers can then derive minimal apps from each app template and integrate TPLUT into newly created apps: one TPLUT per minimal app. This guarantees all apps within a given category share the same configuration (e.g., the same manifest and minimum SDK) and GUI. This systematic approach minimises variability and ensures that the differences measured (e.g., performance, APK size, and permissions) are attributable only to the integrated TPL, not to differences in the base app structure or GUI.

To make the build process as similar as possible to the one developers use to release their apps to the marketplace, we recommend generating and signing the APK file for the minimal app and building its release version. These building and signing are easy tasks for developers because minimal apps are simply apps that integrate the TPLUT.

DEFINING THE SCENARIO TO EXERCISE TPL FUNCTIONALITY. Developers define one playable scenario per TPL to run and exercise the

TPLUT. We assume that developers want to compare TPLs in a category and that these TPLs offer similar features. Thus, developers define a common scenario for all the TPLs under study.

We recommend specifying usage scenarios as shell scripts, which are transferred to and executed on a test smartphone. Once deployed, these scripts simulate user interaction by running the app under test through a sequence of actions (such as `input tap ...` or `input type ...`) and sleep commands. Note that these actions ease the interaction with the phone in the same way app users would do: swipes, text inputs, etc. [203] This allows developers to programmatically interact with their apps. For example, Listing 5.1 shows a simple scenario that runs an app, simulates the user tapping at coordinates (X, Y), and waits for N seconds.

[203] The Android Open Source Project (2007), Android Input.java class

Listing 5.1: Example of scenario.

```
PACKAGE="myPackage"
ACTIVITY="myPackage.MainActivity"

# Launch app
am start -n $PACKAGE/$ACTIVITY

# Tap on coordinate (X, Y)
input tap X Y

# Wait few seconds to simulate user waiting
sleep N
```

Developers can define scenarios on Android phones by enabling developer mode and selecting the input option "Pointer location". This option displays current touch data on the screen, allowing developers to obtain the coordinates of components of the GUI. To execute a scenario, EQuAT uploads it to the phone (`adb push /path/to/scenario.sh`) and executes it (`adb shell nohup sh /path/to/scenario.sh`), using the Android Debug Bridge (ADB).

5.1.1.1 *Measuring Quality Metrics*

Given the app files of minimal apps and a scenario of usage, EQuAT uses a Python script that obtains for each minimal app its size (in MB), the number and type of permissions, the energy consumption (in J), and the CPU, memory, and network usages (in %, MB, and MB, respectively). To collect performance metrics, EQuAT runs each minimal app and plays the defined scenario several times, recording performance measurements. EQuAT automates the metrics measurement process without introducing variations in execution time due to user fatigue or skill level.

Because performance metrics may vary across runs, to obtain statistically meaningful results, we recommend running each minimal

app at least 30 times. EQuAT runs each scenario by default 30 times, though developers can choose a different number of runs.

Next, we describe the quality metrics of apps EQuAT collects:

- APK size (in MB) of each app file. It is the total size of a minimal application after a TPL has been successfully integrated and the app has been built.
- Number and type of permissions, using the command `aapt d permissions ...` available in the Android SDK. This command retrieves all declared Android application permissions from the APK's manifest. This includes standard permissions, as well as special, custom, or library-defined permissions.
- Energy consumption, using a hardware-based approach utilising the Monsoon Low Voltage Power Monitor (LVPM) [148]. Monsoon offers a Python library that allows us to control the power monitor. We power a mobile phone via the LVPM directly connected to the phone's motherboard, bypassing the phone's battery to avoid interference. The power monitor is configured to supply the phone with the constant voltage its battery would provide in a normal setting, $4.4V$ in our case. The electric current drawn by the phone is then registered by the power monitor with a sampling frequency of $\approx 5kHz$.

  [148] Monsoon Solutions, Inc. (2015), Power Monitor Software

  To ensure accurate energy measurements, the phone is not directly connected to the computer via USB, as this would power the phone and bypass the LVPM, leading to corrupted data. Instead, the USB connection passes through a power monitor that turns it off during experiments to prevent this issue. The total energy consumption is the sum of the energy associated with each sample.

  Note that, although there exist software-based approaches to measure the energy consumption of apps, as PETRA [48], they rely on different sources of information to estimate the measurements, which are not guaranteed to be of high absolute accuracy. Thus, we decided to use a hardware-based approach.

  [48] Di Nucci et al. (2017), "Software-Based Energy Profiling of Android Apps: Simple, Efficient and Reliable?"
- CPU usage, running in the background, the `top` command periodically (at one-second intervals) on the phone, filtering by the name of the process associated with an app. It measures CPU usage as a percentage (%), indicating how much of the processor's capacity the system is currently using.
- Memory usage, running in background the Android command `dumpsys meminfo` on the phone. It uses the Proportional Set Size (PSS), as the Android documentation suggests, to estimate the

app's RAM use, including shared pages across processes [67]. All RAM pages that are unique to a process directly contribute to its PSS value, while pages that are shared with other processes contribute to the PSS value only in proportion to the amount of sharing.

[67] Google (2016), Investigating Your RAM Usage

- Network usage, using the tool `tcpdump` [66] on the phone via the Android command `adb` to capture the number of bytes transmitted over the network connection. Network usage refers to the amount of data moving across a network (Wi-Fi, 3G, 4G...).

[66] Google (2015), Android tcpdump

5.1.1.2 *Comparing the Quality Metrics*

Once EQuAT collects quality metrics for N runs, it uses an R script that aggregates all the values of each TPL using the median function. The aggregation uses the median because it is a robust measure of central tendency, while the mean is not. Then, it applies statistical tests to determine whether the differences in quality metrics between each pair of TPLs are significant.

Following the evolution of statistical practices in empirical software engineering [237], EQuAT adopts a non-parametric by default strategy. While parametric tests are more powerful when normality and homoscedasticity assumptions are met, software performance data is frequently skewed and contains outliers that violate these assumptions. Thus, rather than performing automated normality tests, EQuAT performs the pairwise comparisons between different TPLs using the Wilcoxon rank-sum test [86], with a confidence level of 95% ($\alpha = 0.05$). The null hypothesis is that there is no systematic difference between the distributions of the two compared TPLs. When a comparison is statistically significant (p-value < 0.05), EQuAT quantifies the magnitude of the difference using Cliff's δ [38]. This non-parametric strategy is more conservative than parametric alternatives for the proposed approach, significantly reducing the risk of Type I errors (false positives) that occur when normality assumptions are violated.

[237] de Oliveira Neto et al. (2019), "Evolution of statistical analysis in empirical software engineering research: Current state and steps forward"

[38] Cliff (1996), Ordinal Methods for Behavioral Data Analysis

As output, EQuAT reports quality metrics for each TPL into two different plots. One plot shows the APK sizes, the number of permissions, and the distributions of performance metrics (energy consumption and CPU, memory, and network usage) for each TPL. A second plot shows the pairwise comparison of the TPLs under consideration for each quality metric. It includes the metric value of each TPL, the difference of the metric values, whether differences are significant, and, in that case, the magnitude of the effect size. EQuAT also computes the TPLs ranking for each quality metric. First, it aggregates the data for each TPL and metric, computing the median over all the independent runs. Second, it obtains the ranking of each TPL for each quality metric using aggregated data. With this information, developers can determine which TPL is best for each quality metric.

5.1.2 *Validation*

In this Section, we validate EQuAT through a case study. Our hypothesis is that TPLs offering similar functionalities have a different impact on apps' quality: (1) sizes, (2) numbers and types of requested permissions, (3) energy consumption, (4) CPU usage, (5) memory usage, and (6) network usage. To illustrate how developers can leverage EQuAT in practice, we consider the following scenario. A development team is preparing a new mobile app and needs to integrate TPLs for crash-reporting and monitoring. In addition, the development team wants to get some revenue by including a TPL for advertising. There are multiple TPL candidates for crash-reporting, monitoring, and advertising. However, the team wants to select options that provide the required functionality while maintaining high performance and minimising risks to privacy, security, and regulatory compliance. The team proceeds as follows:

- Identify candidate TPLs for each functional category. This is done manually by searching online, reviewing references, or consulting specialised developer forums.
- Generate the app template for each TPL category and derive a minimal app and its corresponding scenario for each TPL under evaluation. These scenarios provide representative usage patterns needed for accurate measurement.
- Run EQuAT to extract permissions, APK size, and runtime energy consumption and CPU, memory, and network usages.
- Compare overall quality and performance implications to understand the broader impact of each TPL. EQuAT provides a consistent basis for comparing all these dimensions across TPLs.
- Assess permission types and protection levels. EQuAT reports the list of permissions requested by each TPL. The developer then evaluates their protection level –*Normal*, *Dangerous*, *Signature*, or custom OS-level permissions– using internal policies or the Android documentation.
- Make an informed selection balancing functionality, performance, and risk. Based on the metrics, the team may decide to: prefer an advertising TPL with low energy consumption if energy or even CPU overhead is critical; select a monitoring library that minimises network usage; adopt a crash-reporting tool that adds minimal app size.
- Document decisions for compliance and maintainability. Since EQuAT outputs structured, reproducible results, the team can record the metrics used for comparison, the justification for

Table 5.1: Popular Android TPLs for popular TPL categories.

Name	Version	Provider	Category
Admob	20.5.0	Google	Advertising
Chartboost	9.1.0	Chartboost	Advertising
Max	11.5.3	AppLovin	Advertising
ACRA	5.9.7	ACRA	Crash-reporting
Crashlytics	29.0.4	Google	Crash-reporting
NewRelic	6.4.1	New Relic	Crash-reporting
Amplitude	2.23.2	Amplitude	Monitoring
Firebase	29.0.4	Google	Monitoring
NewRelic	6.4.1	New Relic	Monitoring

the decisions made, and any identified privacy or performance risks. This is particularly valuable for audits and compliance with regulations such as the General Data Protection Regulation (GDPR).

Next, we describe the objects of study (popular TPLs for advertising, crash-reporting, and monitoring categories), the procedure to run EQuAT, and the obtained results.

5.1.2.1 *Objects*

AppBrain, a statistics Web platform, automatically analyses Android apps on Google Play and retrieves all used libraries by analysing the APK files. Allegedly updated daily, their website reports on the number of published applications, trending apps, download and rating statistics, and the popularity of Android libraries. Based on the Android library stats page [12], we select three of the most popular categories of TPLs and, for each of these categories, three TPLs used by Android developers. We summarise this information in Table 5.1, where we also show the version and provider of each TPL.

Advertising TPLs allows developers to keep their content free and available while still generating revenue. Analytic TPLs provides insights that help developers understand how their users are behaving, where they might be losing them, and more. Crash-reporting TPLs creates detailed reports of errors in apps, which are reported to developers.

5.1.2.2 *Procedure*

We created an app template tailored for each TPL category under study. These app templates are built to the minimum requirements necessary to integrate and execute the TPLUT. For advertising TPLs, the app template has a banner at the top of the device screen to load and show ads. For analytic TPLs, the app template defines a fragment pager

adapter in the main activity with five fragments, each containing a different image and a string identifier (Image1, Image2, Image3, Image4, and Image5). Every time a user swipes left or right, the previous or next fragment is loaded, respectively, and an event is sent to the corresponding TPLUT server. Finally, for crash-reporting TPLs, the app template includes a button that throws a runtime exception to simulate a crash when clicked. From app templates, we derive a stand-alone minimal application for each TPLUT. Thus, minimal apps for TPLs in a category share the same GUI and functionality.

We then define one playable scenario per TPLUT, which specifically exercises the TPL core functionalities intended for integration into a final application. For advertising TPLs, we give libraries 40 seconds in the main activity to load an ad. For analytic TPLs we navigate through the different fragments, then leave the app and wait for 12 seconds, for libraries to update use reports to respective back-ends. Finally, for crash-reporting TPLs, we click the button to throw a runtime exception, then wait 12 seconds for crash reports to be sent to servers.

Once the minimal apps have been developed and the scenarios defined, the corresponding APK file is then built for each minimal app. We then apply EQuAT, introduced in Section 5.1. It automatically runs minimal apps and their corresponding scenarios while collecting quality metrics to compare TPLs. To collect performance metrics, the approach runs each minimal app and plays the defined scenario N times independently. Finally, EQuAT aggregates all the data and generates multiple plots to facilitate comparison of TPLs.

We set the default value of N to 30, which is a conventionally accepted threshold in computer science performance evaluation [62]. We clarify that this value does not inherently guarantee statistical significance. However, it is a rationale-based default intended to provide sufficient observations for non-parametric tests to achieve acceptable statistical power. It provides a balance between achieving statistically significant results and minimising the overhead of the data collection process, which can be time-consuming. Despite the previous, this value is not a fixed constant of the approach, but rather a configurable parameter that developers can adjust.

[62] Georges et al. (2007), "Statistically rigorous Java performance evaluation"

For the case study, we use a Samsung Galaxy A01 Core (also known as Galaxy M01 Core in India and as Galaxy A3 Core in Africa). It is a budget Android smartphone manufactured by Samsung Electronics as part of its A series. It was announced in July 2020 and launched in August 2020 as a low-cost phone. It features a 5.3-inch (14.6 mm) 720p display and runs One UI Core 2.0 on top of Android 10 Go Edition. It is a smaller and lighter version of the Samsung Galaxy A01.

5.1.2.3 *Android Permission Types and Protection Levels*

In addition to collecting a number of permissions for TPLs, we also analyse their protection levels. Android [76] employs a permission

system to control access to sensitive resources and user data. Permissions are categorised into different protection levels, including *Normal* permissions (which pose minimal risk to user privacy) and *Dangerous* permissions (which grant access to sensitive information, e.g., camera, location, contacts, etc., and require the user's explicit consent). We study the protection level of permissions introduced by TPLs on apps. We focus on dangerous permissions because they have the most significant impact on user privacy.

5.1.2.4 *Results*

Once EQuAT collects quality metrics of TPLs, it shows the distribution of each quality metric for each TPL. Fig. 5.2 shows the distribution of each quality metric and TPL, grouped by TPL category. As expected, TPLs differ in their quality metric values. For advertising TPLs, Admob increases both the APK size and the number of required permissions. Chartboost requires fewer permissions and, on average, uses less CPU, memory, and energy than Admob. Even though it requires more permissions than Chartboost, Max seems better overall across the previously mentioned system metrics. All three libraries have comparable network usage. Concerning the crash-reporting category, ACRA increases the APK size and increases network usage compared to the other TPLs in that category; however, it requires fewer permissions. Finally, for the monitoring category, energy consumption, CPU, memory, and network usages are very similar among Amplitude, Firebase, and NewRelic. Nonetheless, Firebase stands out for its larger APK size and greater number of permissions. We discuss permissions qualitatively at the end of this Section.

In addition to the distribution of metric values, EQuAT also computes the ranking of each TPL category and quality metric. Table 5.2 shows the ranking of subject TPLs. With this information, developers can determine which TPL performs better across different quality metrics. For the advertising TPL category, Admob ranks third for all quality metrics. Max TPL ranks first for APK size, energy consumption, and CPU and memory usage, but it requires more Android permissions and consumes more data than Chartboost, which ranks best in terms of the number of permissions and network usage. Regarding the crash-reporting TPL category, in comparison to ACRA and NewRelic, Crashlytics ranks first for performance metrics (energy consumption, CPU, memory, and network usages). However, Crashlytics ranks second for APK size and number of permissions. Finally, concerning the monitoring TPL category, a trade-off occurs when comparing quality metrics. NewRelic ranks first for energy consumption and CPU usage, but second for APK size, number of permissions, and memory and network usage. Although Amplitude ranks first for APK size, number of permissions, and memory usage, it ranks third for CPU and network usage.

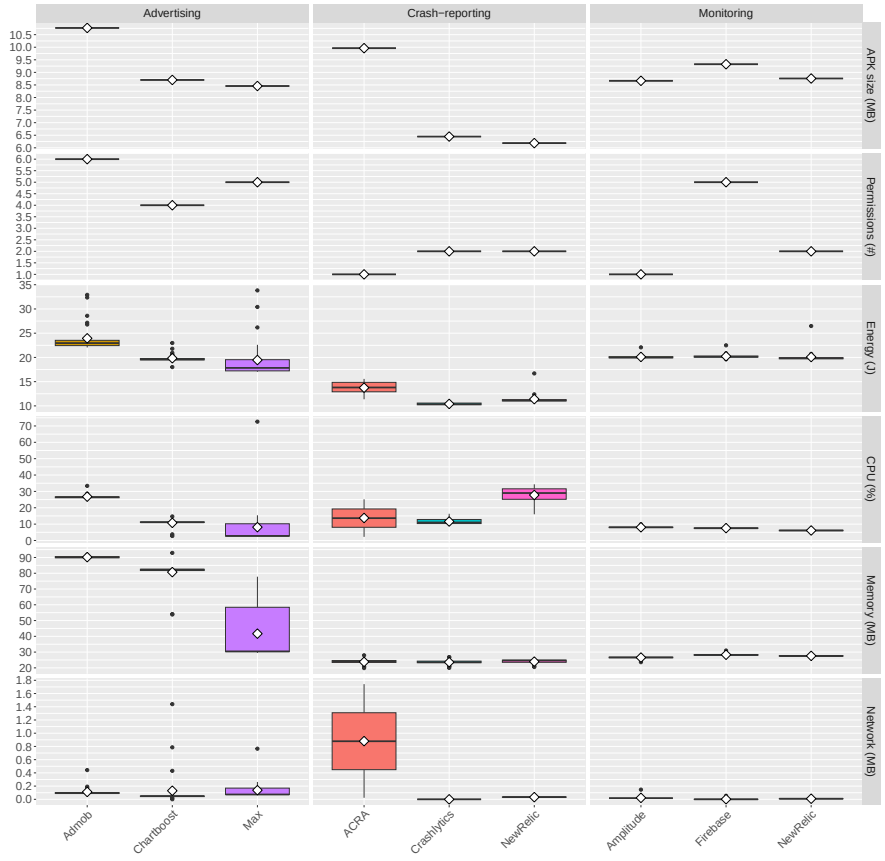


Figure 5.2: Distribution of quality metrics for each TPL under study, grouped by TPL category. The lines in the boxes indicate the minimum value, lower quartile, median, upper quartile, and maximum values. The "◇" symbol represents the average value.

Although rankings are useful for intuitive comparisons, they do not allow informed decision-making because differences between TPLs may be small or even not statistically significant. Fig. 5.3 shows the pairwise comparison of the TPLs under study for each metric. Each cell contains the difference of the medians in %, the magnitude of the difference (in MB, number of permissions, Joules, %, MB, and MB, for APK size, permissions, energy consumption, CPU, memory, and network, respectively), and the magnitude of the effect size (small, medium, or large). The effect size is "NA" when it does not apply (i. e.for APK size and permissions metrics where we only have a measurement for each TPL). Absent values indicate cases where there is no statistically significant difference. Thus, for each cell in the matrix, negative differences (green colours) mean that the TPL at the bottom is better than the TPL at the left. On the contrary, positive differences (red colours) mean that the TPL at the bottom is worse than the TPL at the left. Therefore, greener colours have higher differences in favour of the TPL at the bottom, and redder colours have higher differences in

Table 5.2: Rankings provided by EQuAT for each quality metric and TPL.

TPL	APK size	#Permissions	Energy	CPU	Memory	Network	Category
Admob	3	3	3	3	3	3	Advertising
Chartboost	2	1	2	2	2	1	Advertising
Max	1	2	1	1	1	2	Advertising
ACRA	3	1	3	2	2	3	Crash-reporting
Crashlytics	2	2	1	1	1	1	Crash-reporting
NewRelic	1	2	2	3	3	2	Crash-reporting
Amplitude	1	1	2	3	1	3	Monitoring
Firebase	3	3	3	2	3	1	Monitoring
NewRelic	2	2	1	1	2	2	Monitoring

favour of the TPL at the left. Only cases with a statistically significant difference are shown.

For the advertising TPL category, we observe that Admob increases APK size by 21.46% (2.31MB) and 19.21% (2.07MB) with respect to Max and Chartboost, respectively. Concerning the number of permissions, Chartboost requires 20% (1) and 33.33% (2) fewer permissions than Max and Admob, respectively. We also observe that Max is the most energy-efficient TPL. Admob and Chartboost consumes 22.30% and 9.01% more energy than Max, respectively. Max also performs better than others in terms of CPU and memory usage, with large differences. Admob uses 89.47% and 66.23% more CPU and memory than Max, respectively. Chartboost uses 74.93% and 63% more CPU and memory than Max, respectively. However, Chartboost performs better than others in terms of network usage: it uses 33.33% and 52.36% (100% - 47.64%) fewer data over the network than Max and Admob, respectively.

Concerning the crash-reporting TPL category, ACRA and Crashlytics increase APK size by 37.90% (3.78MB) and 4.05% (0.26MB) with respect to NewRelic. Regarding the number of permissions, ACRA requires half as many as others. However, we observe that ACRA is the least efficient in terms of energy consumption and network usage. It consumes more energy (19.36% and 25.01%) and transmits more data over the network (96.22% and 100%) than NewRelic and Crashlytics, respectively. Regarding CPU usage, ACRA and Crashlytics use 52.92% and 62.07% less CPU than NewRelic, respectively.

For the monitoring TPL category, Amplitude reduces APK size by 1.05% (0.09MB) and 7.06% (0.66MB) with respect to NewRelic and Firebase, respectively. Regarding the number of permissions, Amplitude requires 50% (1) fewer permissions and 80% (4) fewer permissions than NewRelic and Firebase. In terms of energy consumption and CPU usage, NewRelic performs better than others. Amplitude and Firebase consume slightly more energy (0.78% and 1.53%) and use more CPU (23.91% and 18.07%) than NewRelic, respectively. Amplitude uses 3.38% and 5.71% less memory than NewRelic and Firebase,

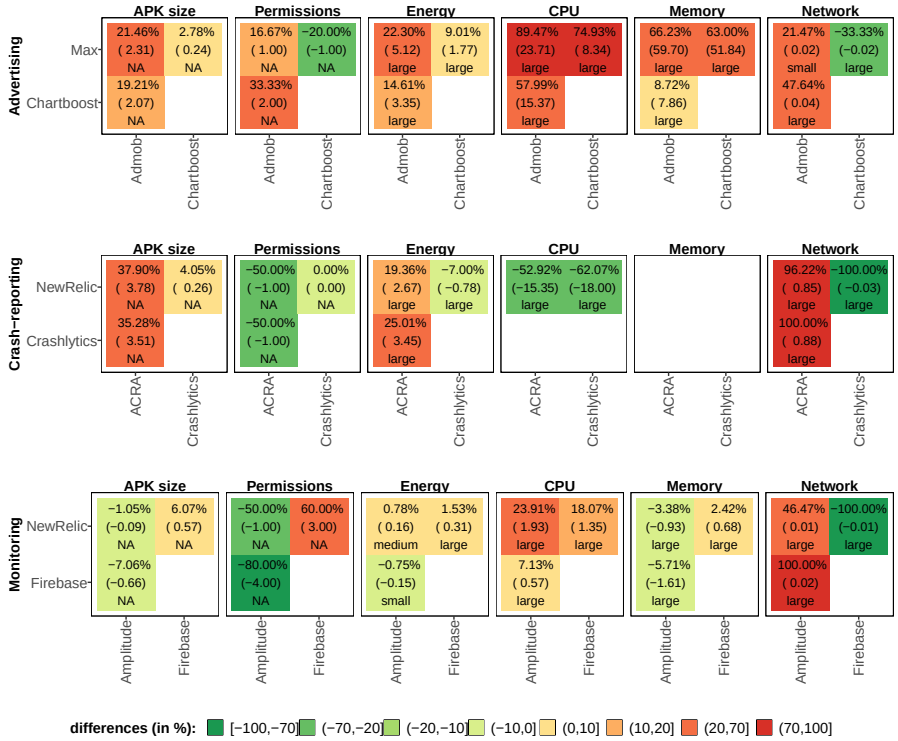


Figure 5.3: Pairwise comparison of TPLs and quality metrics. Each cell contains the difference of the medians in %, the magnitude of the difference (in MB, number of permissions, Joules, %, MB, and MB, for APK size, permissions, energy consumption, CPU, memory, and network, respectively), and the magnitude of the effect size (small, medium, or large). The effect size is "NA" when it does not apply (i.e., for APK size and permissions metrics where we only have a measurement for each TPL). Absent values indicate cases where there is no statistically significant difference. For each cell in the matrix, negative differences (green colours) mean that the TPL at the bottom is better than the TPL at the left. On the contrary, positive differences (red colours) mean that the TPL at the bottom is worse than the TPL at the left. Therefore, greener colours have higher differences in favour of the TPL at the bottom and redder colours have higher differences in favour of the TPL at the left.

respectively. However, Amplitude transmits more data over the network (46.47% and 100%) than NewRelic and Firebase, respectively.

Next, we study the protection level of permissions introduced by TPLs on apps. Table 5.3 presents the Android permissions and their protection levels for the TPLs under study, illustrating the varying levels of risk associated with different permission types. The results demonstrate that integrating TPLs generally increases the risk associated with the requested permission types. Notably, libraries like Max and Firebase introduce two *Dangerous* permissions, whereas Admob introduces only one. The overall trend indicates that incorporating

Table 5.3: Android permissions and their protection level for the TPLs under study.

TPL	Category	Permission	Protection level
Admob	Advertising	android.permission.INTERNET	Normal
Admob	Advertising	android.permission.ACCESS_NETWORK_STATE	Normal
Admob	Advertising	com.google.android.gms.permission.AD_ID	Dangerous
Admob	Advertising	android.permission.WAKE_LOCK	Normal
Admob	Advertising	android.permission.RECEIVE_BOOT_COMPLETED	Normal
Admob	Advertising	android.permission.FOREGROUND_SERVICE	Normal
Chartboost	Advertising	android.permission.INTERNET	Normal
Chartboost	Advertising	android.permission.ACCESS_NETWORK_STATE	Normal
Chartboost	Advertising	android.permission.READ_PHONE_STATE	Dangerous
Chartboost	Advertising	com.google.android.gms.permission.AD_ID	Dangerous
Max	Advertising	android.permission.INTERNET	Normal
Max	Advertising	android.permission.ACCESS_NETWORK_STATE	Normal
Max	Advertising	com.google.android.gms.permission.AD_ID	Dangerous
Max	Advertising	com.applovin.array.apphub.permission.BIND_APPHUB_SERVICE	Dangerous
Max	Advertising	android.permission.ACCESS_WIFI_STATE	Normal
ACRA	Crash-reporting	android.permission.INTERNET	Normal
Crashlytics	Crash-reporting	android.permission.INTERNET	Normal
Crashlytics	Crash-reporting	android.permission.ACCESS_NETWORK_STATE	Normal
NewRelic	Crash-reporting	android.permission.INTERNET	Normal
NewRelic	Crash-reporting	android.permission.ACCESS_NETWORK_STATE	Normal
Amplitude	Monitoring	android.permission.INTERNET	Normal
Firebase	Monitoring	android.permission.INTERNET	Normal
Firebase	Monitoring	android.permission.ACCESS_NETWORK_STATE	Normal
Firebase	Monitoring	android.permission.WAKE_LOCK	Normal
Firebase	Monitoring	com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE	Dangerous
Firebase	Monitoring	com.google.android.gms.permission.AD_ID	Dangerous
NewRelic	Monitoring	android.permission.INTERNET	Normal
NewRelic	Monitoring	android.permission.ACCESS_NETWORK_STATE	Normal

TPLs can lead to applications requiring more permissions, potentially raising concerns about user privacy and security.

• • •

We have seen how EQuAT allows comparison of Android libraries, helping developers choose the most efficient one for a given use case. Let us now explore in depth the energy consumption of applications and discover how it can be distributed across smartphone components.

5.2 UNDERSTAND MOBILE HARDWARE CONSUMPTION REPARTITION

Traditionally, developers can evaluate the impact of their apps on the power draw of the phone using the Android battery discharge API, which presents a coarse granularity (returning battery percentage values), or an external power measurement solution such as Monsoon LVPM [148], which requires a manual bypass of the battery (difficult with recent smartphones featuring a glued battery) and provides the power consumption of the entire smartphone.

In Android 10 (2019), Google added software support to get power consumption information through the On Device Power Rails Monitor (ODPM) (power rails being a set of hardware probes located down-

stream from the battery, just before hardware components), thus providing finer-grained data than previous battery counters; In 2021, Google released the Pixel 6, the first smartphone to feature the hardware required for ODPM to work.

In this work [79], with Guégain et al., we *(i)* explore using system-level indicators to evaluate the overall smartphone power consumption, comparing different statistical learning models, we *(ii)* leverage ODPM data as ground truth to evaluate the pertinence of models, and we *(iii)* explore the impact of the training dataset size on the precision of the power model.

[79] Guégain et al. (2025), "AndroWatts: Unpacking the Power Consumption of Mobile Device's Components"

Table 5.4: The monitored metrics and their Pearson Correlation Coefficient to the component they represent (All correlations are significant, $p < 0.01$).

Metrics	Description	Unit	Proxy of	Correlation to proxy
`Discharge_rate`	Average battery discharge rate	μW	-	-
24 Power Rails	ODPM metrics for the 24 traced component	μW	-	-
`RedLvl`	Average level of red on the screen during the test	$[0-255]$	Display	0.16
`GreenLvl`	Average level of green on the screen during the test	$[0-255]$	Display	0.29
`BlueLvl`	Average level of blue on the screen during the test	$[0-255]$	Display	0.54
`Brightness`	Brightness of the screen	$[1-100]$	Display	0.63
`CPU_LITTLE_FREQ`	Average frequency for the little CPU group	KHz	CPU little	0.92
`CPU_MID_FREQ`	Average frequency for the medium CPU group	KHz	CPU mid	0.64
`CPU_BIG_FREQ`	Average frequency for the big CPU group	KHz	CPU big	0.93
`GPU0_FREQ`	Average frequency for the GPU0	KHz	GPU	0.96
`GPU1_FREQ`	Average frequency for the GPU1	KHz	GPU	0.99
`GPU_MEM`	GPU memory usage	Bytes	GPU	0.47
`Wi-Fi_data`	Total of data emitted and received	Bytes	Wi-Fi antenna	0.65

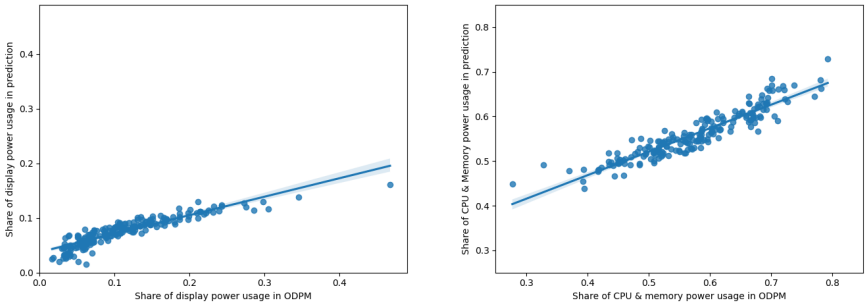
USING SYSTEM-LEVEL METRICS TO BUILD A POWER MODEL. Before using ODPM at all, we ensure that system-level metrics, such as screen brightness and CPU usage, can be used to estimate a smartphone's global power consumption. To do so, we create an Android app that can individually stimulate smartphone components using associated metrics (listed in Table 5.4) and, after running an execution scenario, provides usage metrics that can be retrieved from the phone. We use this application with 1,000 randomly generated configurations to create a dataset of experiments containing the phone's instantaneous power consumption and the values of the selected metrics.

Analysing this dataset allows us to find out which metrics would be meaningful to our power model: in particular, all CPU and GPU seem significantly correlated to the battery discharge rate (≥ 0.53), as opposed to screen metrics, either colour levels or brightness (even though screen brightness is correlated to the screen power draw).

We then compare different statistical learning models to predict the battery discharge rate from its state (derived from system-level metrics) and find that linear models achieve an R^2 of 0.91, indicating that the

Table 5.5: Accuracy of per-component predictions (all correlations are significant, $p < 0,01$).

Rail group	Av. share (%) (measured)	Av. share (%) (estimated)	Av. error (%) (relative)	Av. error (%) (Absolute)	CV (%) (ground truth)	Correlation (Spearman's ρ)
Display	11,17	7,54	−20,48	29,29	61,48	0,94
CPU & Memory	57,55	57,22	0,83	7,00	16,19	0,96
Wi-Fi	8,92	0,00	−100,00	100,00	31,23	-
GPU	9,24	7,11	−16,44	24,85	34,36	0,92



(a) Predicted vs. measured share of the display power usage.

(b) Predicted vs. measured share of the CPU & Memory power usage.

Figure 5.4: Comparison of Predicted vs. measured share of power usage for the assessed component groups.

battery discharge rate can be largely explained by the smartphone state.

SYSTEM-LEVEL METRICS VS ODPM DATA. Power draw of a given component can be expressed as the sum of $X_1, X_2, \ldots, X_n$ proxy metrics with associated $\beta_1, \beta_2, \ldots, \beta_n$ coefficients:

$$E_C = \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n \tag{5.1}$$

While this allows us to compare a component's predicted draw with the current draw reported by ODPM, it does not allow for a precise estimation of the battery draw rate, as the sum of all ODPM rails does not match the actual battery discharge rate. Additionally, some power rails interact with one another: both GPU rails are perfectly correlated, and the INFRASTRUCTURE rail is strongly correlated with both memory and one CPU rail, hindering our model's ability to differentiate the power draw of some components.

We acknowledge this coarse-grain approach, and after regrouping some metrics together, we apply Equation 5.1 to a linear model using metrics presented in Table 5.4, and present results in Table 5.5; Figure 5.4 also displays a comparison of predicted and measured share of two rail groups.

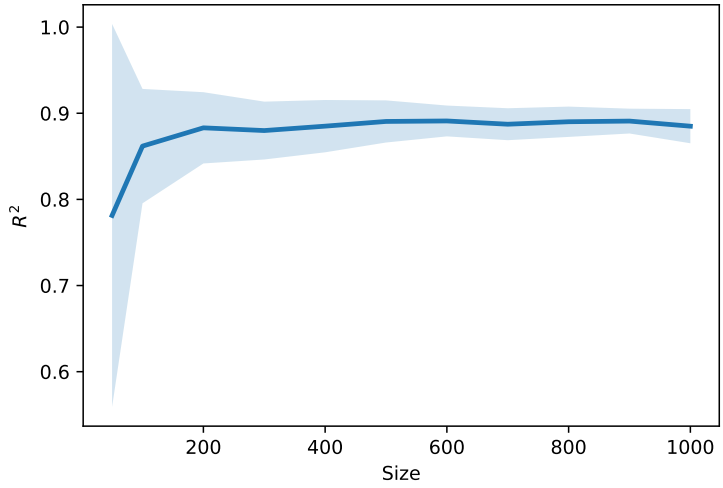


Figure 5.5: The evolution of R^2 according to the size of the dataset.

Both the figure and the table show that the power model achieves good predictive performance, with strong correlation between estimated and ground-truth power draw across all rail groups, even though it underestimates power consumption in all cases. The Wi-Fi group is a notable exception, being negatively correlated with the battery discharge speed and the CPU rail; we suspect CPU-intensive smartphone use has a throttling effect, reducing the network traffic passing through the Wi-Fi antenna.

TRAINING DATASET SIZE IMPACT ON MODEL PRECISION. Now that we have strong confidence in our power model, let's explore how the size of the training dataset affects its precision and whether we can reduce it without affecting model performance. Indeed, building the training dataset and using it for measurements takes a lot of resources, both in disk space and time: the original 1,000-experiment took ≈ 26.7 hours to run, producing 256 gigabytes of data that took 3 hours to parse.

To study whether data size could be reduced without compromising the precision of the power model, we train a linear model on subsets of the original dataset, ranging from 50 to 1,000 samples. Figure 5.5 describes the precision of the linear model compared to the number of experiments included in the training dataset. It shows that the power model's accuracy plateaus at ≈400 measurements, meaning that beyond that point, additional experiments are not required to improve its performance.

5.3 DISCUSSION

5.3.1 *On the usage of EQuAT*

Android developers often integrate TPLs into their apps to implement specific functionality. From the case study, we observed that TPLs impacts the quality of Android apps. Thus, EQuAT could potentially provide developers a mechanism to know the impact of TPLs on their apps. However, there is a trade-off between different TPLs and their impact on apps' quality metrics.

> We suggest that developers select a TPL according to the specific context of use experienced by their users.

EMERGING MARKETS EXAMPLE. Let us focus, for instance, on emerging markets. Users in emerging markets own low-end devices with very limited CPU and memory; in addition, a majority of users in emerging markets face limited and expensive access to data connections. [74] Thus, it is important to prioritise APK size and CPU, memory, and network usages in this context.

Based on our validation, Max is the most advisable TPL for advertising in apps targeting emerging markets, being the TPL that minimises the APK size and is the most efficient in terms of energy consumption and CPU and memory usages. Using Max to show ads instead of other advertising TPLs, developers can reduce app size by up to 21% and improve energy consumption and CPU and memory usage by up to 22%, 89%, and 66%, respectively. However, Chartboost transmits 33% less data over the network than Max. Regarding the crash-reporting TPL category, NewRelic reduces the APK size by up to 38% while using more CPU than others.

Although Crashlytics increases APK size by 4% compared to NewRelic, it consumes less energy and uses less CPU and network than the others (differences in memory usage are not significant). Thus, Crashlytics could be advised for crash-reporting when targeting emerging markets.

Concerning the monitoring TPL category, Amplitude minimises the APK size and memory usage, at the cost of increasing CPU and network usage by up to 24% and 100%, respectively, compared to others TPLs in the same category.

PERMISSION TYPES. While prior studies often focus on the number of permissions requested by an app or its TPL, it is equally important to analyse the type of permission required. Not all permissions pose the same level of risk. *Normal* permissions allow access to low-impact features and do not require explicit user approval, whereas *Dangerous* permissions grant access to sensitive user data, such as device identifiers or location information, and must be explicitly granted by

the user. Consequently, a single *Dangerous* permission can introduce significantly more risk than several *Normal* permissions.

Moreover, certain permissions may have privacy and security implications that extend beyond their immediate functionality. For example, `READ_PHONE_STATE` is a dangerous permission that provides access to sensitive phone state information, including cellular network details, ongoing call status, and registered phone accounts. This permission is used by Chartboost. Additionally, several TPLs, such as Admob, Chartboost, Firebase, and Max, require non-standard Android OS-Level permissions such as `com.google.android.gms.permission.AD_ID`. This permission, defined within Google Play Services, is specific to Google's ecosystem, and since Android 12, Google requires apps that use the Advertising ID to explicitly declare this permission in their `AndroidManifest.xml`. Google also enforces policies restricting its use, particularly in response to privacy regulations like GDPR and CCPA [71].

[71] Google (2021), Advertising ID

In the context of this work, EQuAT directly supports this type-based permission analysis. EQuAT relies on the Android SDK tools to extract the complete list of permissions requested by each TPL. Android SDK tools provide the raw permission names exactly as declared in the TPL's manifest. Once these permissions are collected, the responsibility of determining their protection level lies with the developer or analyst using EQuAT. This design ensures flexibility, allowing the app developer to interpret risks in accordance with the latest Android documentation, regulatory requirements, or project-specific privacy considerations. By exposing the full set of requested permissions and enabling developers to cross-reference their protection levels, EQuAT facilitates a clearer understanding of the associated security and privacy risks introduced by TPLs.

Thus, understanding the type of permissions requested by TPLs is critical for several reasons:

- Risk assessment. Different permission types introduce varying levels of security and privacy risks. Analysing the type helps quantify potential threats more effectively than merely counting permissions.
- Compliance considerations. With data protection laws such as the GDPR and the California Consumer Privacy Act (CCPA), permissions that grant access to user data may raise regulatory concerns.
- User awareness and trust. App users often grant permissions without fully understanding their implications. Highlighting the type of permissions requested can improve transparency and user trust.
- Third-party influence on app security. Developers integrating TPLs may unintentionally introduce permissions that could com-

promise user privacy. Analysing the type of permissions can help developers make informed decisions about which TPLs to include in their apps.

All the previous confirms that making decisions without a systematic evaluation of TPLs likely has a negative impact on the quality and performance of apps. For instance, relying solely on the general reputation or popularity of TPLs is not a better proxy for quality. Admob is a very popular TPL for advertising, but, based on our results, it performs worst across all metrics (size, permissions, energy consumption, and CPU, memory, and network usage) and introduces one *Dangerous* permission.

Something similar happens with Firebase, which claims "*Run your app with confidence and deliver the best experience for your users. Launch, monitor, and iterate with AI-assistive tools that help you optimize your app's quality and experience.*" [202]. However, based on our results, Amplitude and NewRelic are more efficient monitoring TPLs in terms of energy consumption. Looking only at APK size is neither recommended, as it ignores runtime performance. Based on our results, although NewRelic minimises the APK size of apps in the crash-reporting category, it is the worst in terms of CPU and memory usage. Something similar applies to the monitoring category: Amplitude minimises the APK size but uses more CPU and network than Firebase and NewRelic.

[202] Tamplin et al. (2012), Firebase services

5.3.2 *Threats to Validity*

Threats to internal validity concern factors that could have influenced our results. We computed quality metrics (performance metrics, APK size, and number of permissions) using well-known approaches. Additionally, we repeated our measurements multiple times to confirm their statistical reliability. All minimal apps shared identical configurations and GUI, ensuring that observed differences can be attributed primarily to the integrated TPL rather than to variation in app structure.

Threats to construct validity concern the relationship between theory and observation and the extent to which the measures represent real values. We used a Samsung Galaxy A01 Core phone. We measured power usage (energy consumption) at a sampling frequency one order of magnitude higher than in prior studies. However, with the physical measurement of energy consumption come the limitations of measurement and experimentation that exist in the natural sciences and engineering [85]. CPU, memory, and network usages were collected using the commands `top` and `dumpsys`, and the tool `tcpdump` on the phone, respectively, which may have introduced extra energy consumption. To avoid any impact from other metrics' measurements, we collected power usage separately. The number of permissions was collected using the command `aapt`, which is available in the Android

[85] Hindle (2016), "Green Software Engineering: The Curse of Methodology"

SDK. Furthermore, our experimental design relies on minimal applications and controlled scenarios to isolate the impact of individual TPLs. While this design enables fair comparison across libraries, it may not capture all interactions present in complex real-world apps. Consequently, the measured metrics represent controlled approximations of real usage rather than exhaustive representations of all deployment contexts.

Threats to external validity concern the generalization of our findings. First, our experiments were conducted on a single device (Samsung Galaxy A01) running Android 10 Go Edition. Therefore, absolute metric values may differ on other devices, hardware configurations, or Android versions. Although we have no reason to believe that relative trends between TPLs would fundamentally change, replication on additional devices would strengthen generalizability. Second, the selection of TPL categories and libraries followed a purposeful, non-random sampling strategy rather than statistical sampling. We selected three widely used TPL categories and, within each category, three popular libraries based on usage statistics reported by AppBrain. This choice aimed to ensure practical representativeness by focusing on libraries frequently encountered by Android developers in practice. This sampling decision reflects the primary goal of our study, which is not to produce an exhaustive catalogue or ranking of all existing TPLs, but rather to validate and demonstrate a systematic methodology for comparing alternative libraries under controlled conditions. Consequently, our results should be interpreted as evidence of the applicability and usefulness of the proposed evaluation approach rather than definitive quality rankings for the entire ecosystem of TPLs.

Threats to conclusion validity concern the relationship between the experiment and the outcome. We conducted our research using popular TPLs and applied appropriate statistical procedures. We paid attention not to violate assumptions of the constructed statistical models. In particular, we used non-parametric tests that do not assume the underlying data distribution.

5.3.3 *Future works*

Despite the technical feasibility demonstrated by EQuAT, several limitations remain. The creation of minimal apps for TPL comparison still requires manual effort, which we aim to mitigate through future automation. Furthermore, our current scope is restricted to app user-centric metrics, excluding developer-facing attributes like maintainability or testability. We also acknowledge that our assumption of TPL overhead being additive in complex apps requires further empirical validation. Finally, since actual developers were not involved in this study, the practical utility and ease of integration of EQuAT from a practitioner's perspective remain to be confirmed.

Moving forward, we leverage these limitations to define concrete research directions. We plan to develop IDE plugins or scriptable generators—potentially leveraging Artificial Intelligence—to automatically generate minimal app templates. We also aim to expand our quality model to incorporate ISO 25010 standards, such as testability and portability, through static analysis. To enhance the paper's practical relevance, future work will involve a user study with professional developers to validate the framework's utility in their daily workflows. Lastly, we intend to integrate our approach with tools like LibRadar or LibD to provide automated suggestions for more efficient TPL alternatives.

5.3.4 *Conclusion*

Android is one of the most popular operating systems for mobile devices. Previous experiments showed that more than 60% of the sub-packages in Android apps are from TPLs. Thus, TPLs account for more than 60% of the code in Android apps. Other studies also found inconsistencies between Android descriptions of the apps and requested permissions due to the usage of TPLs.

Although there are multiple TPLs to use, developers need information on the impact of TPLs on the quality of their apps: app size, number of type and permissions, and performance. In this chapter, we proposed an approach for Evaluating Quality Attributes of TPLs (EQuAT) and analyse the impact of TPLs on app size, permissions, and performance. EQuAT allows developers to make informed decisions about the TPLs to integrate into their apps, and it can be applied by developers at any time during the app development process.

To validate EQuAT, we compared the quality metrics of popular TPLs in three popular TPL categories: advertising, analytics, and crash reporting. The validation of our approach showed a trade-off between TPLs and their impact on different quality metrics.

For the advertising category, Max is the least impacting TPL in APK size, energy consumption, and CPU and memory usages. However, Max performs worse than Chartboost for network usage and requires more permissions. For the crash-reporting category, Crashlytics consumes less energy and uses less CPU, memory, and bandwidth than ACRA and NewRelic. However, Crashlytics increases the APK size and requires more permissions than NewRelic and ACRA, respectively. For the monitoring category, Amplitude and NewRelic have different strengths and weaknesses while Firebase is the most impactful for all considered metrics. Despite the previous, Firebase introduces two *Dangerous* permissions while Amplitude and NewRelic do not.

Thus, we suggest developers make their decision about the TPL to integrate in their apps based on the context of use of their end users.

6 CONCLUSION & PERSPECTIVES

While cloud computing plays an essential role in today's services supplying, it also brings in disadvantages: it reduces the privacy of end users and features a non-negligible environmental impact through its data centres. In this context, mobile computing is a potential solution, recognizing that end user devices hold computing power. However, cloud is often preferred because it is more intuitive to users and more easily controllable to service maintainers. Through this thesis, I have demonstrated that smartphones can play a major role in ICT. The potential gains are substantial, and exploring new methods of local computing can lead to improvements in performance and end user privacy.

6.1 CONTRIBUTION

In my contributions, I have explored several ways of restoring independence to end users, notably in terms of privacy and its implications on both data storage and communication. Combined with a controlled energy consumption, we shown that smartphones can play a vital role in telecommunication.

6.1.1 *Protecting user privacy through local data storage*

Identifying local data storage as a lock, we proposed Fast Linear Interpolation (FLI), a time series compression algorithm relying on linear regression, increasing the storage capacity of smartphones; its extension, Fast Linear Interpolation in n-Dimensions (FLInD), adds support for Multivariate Time Series (MTS). We integrated this storage component into the IN-siTu locAtion proteCTion (INTACT) framework and deployed it on smartphones (Android and iOS) to demonstrate that local execution of attack and defence algorithms is now possible, thereby increasing users' privacy.

6.1.2 *Unlocking local device-to-device communication*

Arguing that all automated communications do not require traversing the Internet, which raises both energy and privacy concerns, we studied the case of Device-to-device (D2D) communication. We first ran a survey asking users about their eventual knowledge of D2D and how, when, and why they use it. We then highlighted a lack of uniformity in the implementation of radio technology across platforms and subsequently proposed VENICE, a D2D cross-platform communi-

cation framework that enables experimentation with communication channels.

6.1.3 *Reducing the energy consumption of mobile devices*

Since smartphones run on batteries, their energy consumption can be a barrier to local computing. Given that Third Party Libraries (TPLs) make up most Android applications, we proposed Evaluating Quality Attributes of TPLs (EQuAT), an evaluation framework that enables developers to compare Android libraries and select the most efficient one. We also examined the hardware side of mobile energy consumption and proposed a power model based on system metrics to identify the distribution of instantaneous energy draw across smartphone components.

6.2 PERSPECTIVES

Finally, I propose perspectives on this work by outlining short-term extensions related to my contributions, and describing long-term objectives, related to the broader scope of eschewing services dependency to the cloud.

6.2.1 *Short-term perspectives*

Instil more privacy.

In Chapter 3, I presented our work on *IN-siTu locAtion proteCTion (INTACT)*. This tool includes both attack mechanisms, such as Divide&Stay (D&S), a mobile-compatible implementation of POI-ATTACK, as well as Location Privacy Protection Mechanisms (LPPMs), such as Promesse [170], thus providing a complete evaluation framework for end users to evaluate their data's privacy before sharing it.

The INTACT framework also comprises, at its core, the FLI storage layer, which drastically increases smartphone storage space. This increased capacity provides the opportunity to implement new *in situ* LPPMs, including anonymity-based mechanisms that were previously server-side due to the need to aggregate location data from many users into a single location.

Stop fighting OSes, start networking.

In Chapter 4, I presented our work on Device-to-device (D2D) communication, including the VENICE framework, which explores channel interoperability. Thanks to this tool, we identify one of the locks preventing full D2D adoption is the tedious integration process: De-

velopers currently have to implement support for each channel and each supported OS, a laborious task due to the ever-changing OS APIs, especially the lack of low-level, D2D system APIs.

Therefore, I believe that proposing a higher-level, cross-OS communication D2D API would allow developers to focus on what really matters: interactions between people. Such a unified API could hide technical disparities across devices, finally enabling seamless communication among everyone.

Watch the energy needs of your app.

In Chapter 5, I presented our work on Android energy consumption, providing tools to evaluate the power draw of both software and hardware components of the ecosystem. Since the power model we built is tied to a specific smartphone, the next step is to extend this work and create a portable power model that can be applied to any Android phone. Additionally, since iOS has a considerable market share in some countries [20], portability of our works to iOS should also be considered, so that no one is left behind.

6.2.2 *Long-term perspectives*

Despite previous concerns about the availability of natural resources to continue digitalising our society (*e.g.*, electrifying the global car fleet) [198], the current hardware shortage of hard drives and memory is directly attributable to the AI industry, which requires loads and loads of GPUs.

[198] Stéphant (2022), Mining rush in the 21st century: how far will the limits be pushed? (French)

At the time of writing, huge models are on the rise (I'm still not over the fact that we are not talking in numbers of GPUs but rather in GW of GPUs); I believe the consumed resources VS use-case AI ratio is not worth it, and we don't know yet if this AI phenomenon is a bubble that will collapse soon.

I do not believe encouraging big tech companies in their self-imposed AI race is valuable to anyone on Earth. Instead, the focus must be on including as many people as possible in communities, ensuring they are not excluded by hardware or software obsolescence.

The cloud is not vital.

While the Internet was a decentralised network at its inception, services are now widely centralised around servers in data centres.

I think this model is problematic, particularly for privacy reasons, and thus believe Information and Communications Technology (ICT) should be deployed as close to user devices as possible: this includes reinventing software so that local deployment is possible [110], deploying user-controlled networks [229] and technical stacks [47].

[110] Kleppmann et al. (2019), "Local-first software: you own your data, in spite of the cloud"

Fight the digital divide.

The vast majority of tech companies only cares about the features added to their applications and the count of people using them, imposing a fast software/hardware update cycle users are forced to follow (*e.g.*, upgrading the minimal SDK version of an Android app forces some users to change their smartphone [150]).

[150] Mosesso et al. (2023), "Obsolescence Paths: living with aging devices"

I argue *"deemed-too-old"* devices are actually still usable, through solutions such as Lineage OS; development efforts must however make software more frugal, and hardware more easily repairable.

In short, rather than encouraging the rise of big, energy-consuming, privacy-devouring software and companies, let's keep the control!

BIBLIOGRAPHY

[1] 3GPP. *Release 12*. Standard v0.2.0. 3GPP, 2015.

[2] 3GPP. *Release 15*. Standard v15.0.0. 3GPP, 2019.

[3] 42matters. *Google Play vs the Apple App Store: App Stats and Trends*. Last accessed March 18th, 2026. 2025. URL: https://42matters.com/stats.

[4] Mahmoud Abbasi, Amin Shahraki, Hamid R Barzegar, and Claus Pahl. "Synchronization Techniques in "Device to Device- and Vehicle to Vehicle-Enabled" Cellular Networks: A survey." In: *Computers & Electrical Engineering* 90 (2021), p. 106955.

[5] Michiel Aernouts, Rafael Berkvens, Koen Van Vlaenderen, and Maarten Weyn. "Sigfox and LoRaWAN Datasets for Fingerprint Localization in Large Urban and Rural Areas." In: *Data* 3.2 (June 2018), p. 13. DOI: 10.3390/data3020013.

[6] Andrew Ahn. *How we're helping people find quality apps and games on Google Play*. Last accessed March 18th, 2026. 2017. URL: https://android-developers.googleblog.com/2017/08/how-were-helping-people-find-quality.html.

[7] Yeslam Al-Saggaf. "The Use of Data Mining by Private Health Insurance Companies and Customers' Privacy: An Ethical Analysis." In: *Cambridge Quarterly of Healthcare Ethics* 24.3 (July 2015). ISSN: 0963-1801, 1469-2147. DOI: 10.1017/S0963180114000607.

[8] Wi-Fi Alliance. *Wi-Fi Easy Connect*. https://www.wi-fi.org/discover-wi-fi/wi-fi-easy-connect. Last accessed on 2023/10/31. 2023.

[9] Yanzhe An, Yue Su, Yuqing Zhu, and Jianmin Wang. "TVStore: Automatically Bounding Time Series Storage via Time-Varying Compression." In: *20th USENIX Conference on File and Storage Technologies (FAST 22)*. Santa Clara, CA: USENIX Association, Feb. 2022, pp. 83–100. ISBN: 978-1-939133-26-7. URL: https://www.usenix.org/conference/fast22/presentation/an.

[10] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. "Geo-indistinguishability: Differential privacy for location-based systems." In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013, pp. 901–914.

[11] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper B. Rasmussen. "Nearby Threats: Reversing, Analyzing, and Attacking Google's 'Nearby Connections' on Android." In: *Proceedings 2019 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2019. ISBN: 978-1-891562-55-6. DOI: 10.14722/ndss.2019.23367.

[12] AppBrain. *Android library statistics*. Last accessed March 19th, 2026. 2018. URL: https://www.appbrain.com/stats/libraries.

[13] Apple Fandom. *iPhone (1st generation)*. Last accessed on Apr 2023. 2023. URL: https://apple.fandom.com/wiki/IPhone_(1st_generation).

[14] Apple Support. *Ultra Wideband Availability*. Last accessed on Apr 2023. 2021. URL: https://support.apple.com/en-us/HT212274.

[15] Apple. *iOS `CLLocationManager` documentation*. https://developer.apple.com/documentation/corelocation/cllocationmanager. Last accessed on April 21st, 2024. 2013.

[16] Apple. *AirPlay*. Last accessed on Apr 2023. 2022. URL: https://www.apple.com/airplay/.

[17] Apple. *Nearby Interactions with U1*. Last accessed on Apr 2023. 2022. URL: https://developer.apple.com/nearby-interaction/.

[18] Hadi Asghari, Michel van Eeten, Johannes M. Bauer, and Milton Mueller. *Deep Packet Inspection: Effects of Regulation on Its Deployment by Internet Providers*. SSRN Scholarly Paper ID 2242463. Rochester, NY: Social Science Research Network, Sept. 2013. DOI: 10.2139/ssrn.2242463.

[19] Aloÿs Augustin, Jiazi Yi, Thomas Clausen, and William Mark Townsley. "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things." In: *Sensors (Basel, Switzerland)* 16.9 (Sept. 2016), p. 1466. ISSN: 1424-8220. DOI: 10.3390/s16091466.

[20] Backlinko. *iPhone vs Android Statistics*. Last accessed April 22nd, 2026. 2025. URL: https://backlinko.com/iphone-vs-android-statistics.

[21] Brian Barrett. "The Biggest iPhone News Is a Tiny New Chip Inside It." In: *Wired* (Dec. 2019). ISSN: 1059-1028.

[22] Barbara A. Bays and Michael J Mc Gowan. *Use of RFID for Tracking Government Property - Proof of Concept/Pilot*. Last accessed on Apr 2023. 2017. URL: https://www.osti.gov/biblio/1456529.

[23] Eugen Berlin and Kristof Van Laerhoven. "An on-line piecewise linear approximation technique for wireless sensor networks." In: *IEEE Local Computer Network Conference*. IEEE. 2010, pp. 905–912.

[24] Ajay Bhardwaj and Samar Agnihotri. "Energy-and spectral-efficiency trade-off for D2D-multicasts in underlay cellular networks." In: *IEEE Wireless Communications Letters* 7.4 (2018), pp. 546–549.

[25] Simon Binder. *Drift library*. https://pub.dev/packages/drift. Last accessed on April 21st, 2024. 2019.

[26] Davis Blalock, Samuel Madden, and John Guttag. "Sprintz: Time Series Compression for the Internet of Things." In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2.3 (Sept. 2018). DOI: 10.1145/3264903.

[27] Bluetooth SIG. *Bluetooth Core Specification*. Standard v 1.0B. Bluetooth SIG, 1999.

[28] Bluetooth SIG. *Bluetooth Technology Protecting Your Privacy*. Last accessed on May 2023. 2015. URL: https://www.bluetooth.com/blog/bluetooth-technology-protecting-your-privacy/.

[29] Bluetooth SIG. *Bluetooth Core Specification*. Standard v 5.0. Bluetooth SIG, 2016.

[30] Bluetooth SIG. *Bluetooth Technology Overview*. Last accessed on Apr 2023. 2023. URL: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/.

[31] Bluetooth SIG. *Understanding Bluetooth Range*. Last accessed on Apr 2023. 2023. URL: https://www.bluetooth.com/learn-about-bluetooth/key-attributes/range/.

[32] Bluetooth SIG. *Bluetooth® Core Specification Version 5.2 - Feature Overview*. Last accessed on Apr, 21st 2026. Nov. 2025. URL: https://www.bluetooth.com/bluetooth-core-6-2-feature-overview/.

[33] Carole Cadwalladr. "'I Made Steve Bannon's Psychological Warfare Tool': Meet the Data War Whistleblower." In: *The Guardian* (Mar. 2018).

[34] Neil Campbell. "QR Code Contact Tracing, Vaccine Passports Becoming New Normal in Europe, America." In: *Vision Times* (Jan. 2021).

[35] Sobin Cc, Vaskar Raychoudhury, Gustavo Marfia, and Ankita Singla. "A Survey of Routing and Data Dissemination in Delay Tolerant Networks." In: *Journal of Network and Computer Applications* 67 (May 2016). ISSN: 1084-8045. DOI: 10.1016/j.jnca.2016.01.002.

[36] Jacqui Cheng. "Over 3 Years Later, "Deleted" Facebook Photos Are Still Online." In: *Ars Technica* (Feb. 2012).

[37] Cisco. *Annual Interet Report (2018–2023) White Paper*. Tech. rep. Cisco, 2020.

[38] N. Cliff. *Ordinal Methods for Behavioral Data Analysis*. Erlbaum, 1996. ISBN: 9780805813333. URL: https://books.google.ca/books?id=bIJFvgAACAAJ.

[39] Marco Conti, Andrea Passarella, and Sajal K. Das. "The Internet of People (IoP): A New Wave in Pervasive Mobile Computing." In: *Pervasive and Mobile Computing* 41 (Oct. 2017), pp. 1–27. ISSN: 1574-1192. URL: https://www.sciencedirect.com/science/article/pii/S1574119217303723 (visited on 07/04/2023).

[40] Diane J. Cook and Sajal K. Das. "Pervasive Computing at Scale: Transforming the State of the Art." In: *Pervasive and Mobile Computing* 8.1 (Feb. 2012), pp. 22–35. ISSN: 1574-1192. URL: https://www.sciencedirect.com/science/article/pii/S1574119211001416 (visited on 07/05/2023).

[41] Stephen Cooper. *Ultra Wideband & You*. WLPC Phoenix. Last accessed on Apr 2023. Mar. 2020. URL: https://www.youtube.com/watch?v=TR-rahy3Y2k.

[42] S. Corson and J. Macker. *Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*. Tech. rep. RFC2501. RFC Editor, Jan. 1999, RFC2501. DOI: 10.17487/rfc2501.

[43] Court of Justice of the European Union. *Judgment on Traffic Log Retention in the Face of National Security*. Court judgment ECLI:EU:C:2020:791. Oct. 2020.

[44] Thomas Crowther et al. "Mapping tree density at a global scale." In: *Nature* advance online publication (Sept. 2015). DOI: 10.1038/nature14967.

[45] David Curry. *Android Statistics*. Last accessed March 18th, 2026. 2026. URL: https://www.businessofapps.com/data/android-statistics/.

[46] Lauric Desauw, Adrien Luxey-Bitri, Rémy Raes, Romain Rouvoy, Olivier Ruas, and Walter Rudametkin. "A Critical Review of Mobile Device-to-Device Communication." In: *Proceedings of the 25th International Conference on Distributed Applications and Interoperable Systems (DAIS'25)*. Ed. by Daniel Balouek and Ibéria Medeiros. LNCS. Springer, June 2025, pp. 1–24. ISBN: 978-3-031-95728-4. DOI: 10.1007/978-3-031-95728-4_1. URL: https://inria.hal.science/hal-04198528v1.

[47] Deuxfleurs. *The Garage team - An S3 object store so reliable you can run it outside datacenters*. Last accessed on April 27th, 2026. 2022. URL: https://garagehq.deuxfleurs.fr/.

[48] Dario Di Nucci, Fabio Palomba, Antonio Prota, Annibale Panichella, Andy Zaidman, and Andrea De Lucia. "Software-Based Energy Profiling of Android Apps: Simple, Efficient and Reliable?" In: *Proceedings of the 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. Klagenfurt, Austria, 2017, pp. 103–114.

[49] Roger Dingledine, Nick Mathewson, and Paul Syverson. *Tor: The Second-Generation Onion Router*. Tech. rep. Fort Belvoir, VA: Defense Technical Information Center, Jan. 2004. DOI: 10.21236/ADA465464.

[50] Vivien Dollinger and Markus Junginger. *ObjectBox database*. https://objectbox.io. Last accessed on April 21st, 2024. 2014.

[51] Klaus Doppler, Cássio B. Ribeiro, and Jarkko Kneckt. "Advances in D2D Communications: Energy Efficient Service and Device Discovery Radio." In: *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*. Feb. 2011. DOI: 10.1109/WIRELESSVITAE.2011.5940857.

[52] X. Du, Z. Yang, J. Lin, Y. Cao, and M. Yang. "Withdrawing is believing? Detecting Inconsistencies Between Withdrawal Choices and Third-party Data Collections in Mobile Apps." In: *2024 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2024, pp. 14–14. DOI: 10.1109/SP54263.2024.00014. URL: https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00014.

[53] Cynthia Dwork. "Differential privacy: A survey of results." In: *International conference on theory and applications of models of computation*. Springer. 2008, pp. 1–19.

[54] Jeremy Elson, Lewis Girod, and Deborah Estrin. "Fine-grained network time synchronization using reference broadcasts." In: *ACM SIGOPS Operating Systems Review* 36.SI (2002), pp. 147–163.

[55] Shih-Hau Fang, Hao-Hsiang Liao, Yu-Xiang Fei, Kai-Hsiang Chen, Jen-Wei Huang, Yu-Ding Lu, and Yu Tsao. "Transportation modes classification using sensors on smartphones." In: *Sensors* 16.8 (2016), p. 1324.

[56] Wei Feng, Zheng Yan, Laurence T. Yang, and Qinghua Zheng. "Anonymous Authentication on Trust in Blockchain-Based Mobile Crowdsourcing." In: *IEEE Internet of Things Journal* (2020). ISSN: 2327-4662. DOI: 10.1109/JIOT.2020.3018878.

[57] Michelle Fleury and Nathalie Jimenez. *'I can't drink the water' - life next to a US data centre*. Last accessed on April 28th, 2026. 2025. URL: https://www.bbc.com/news/articles/cy8gy7lv448o.

[58] GSMA. *Over Half World's Population Now Using Mobile Internet*. Last accessed on Apr 2023. 2021. URL: https://www.gsma.com/newsroom/press-release/over-half-worlds-population-now-using-mobile-internet/.

[59] Alex Galakatos, Michael Markovitch, Carsten Binnig, Rodrigo Fonseca, and Tim Kraska. "FITing-tree: A data-aware index structure." In: *Proceedings of the 2019 International Conference on Management of Data*. 2019, pp. 1189–1206.

[60] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. "De-anonymization attack on geolocated data." In: *Journal of Computer and System Sciences* 80.8 (2014), pp. 1597–1614.

[61] Gabriel K. Gegenhuber, Philipp É. Frenzel, Maximilian Günther, Johanna Ullrich, and Aljosha Judmayer. "Hey there! You are using WhatsApp: Enumerating Three Billion Accounts for Security and Privacy." In: *33rd Annual Network and Distributed System Security Symposium, (NDSS) 2026, San Diego, California, USA, February 23–27, 2026*. The Internet Society, 2026. DOI: https://dx.doi.org/10.14722/ndss.2026.230805. URL: https://github.com/sbaresearch/whatsapp-census.

[62] Andy Georges, Dries Buytaert, and Lieven Eeckhout. "Statistically rigorous Java performance evaluation." In: *SIGPLAN Not.* 42.10 (Oct. 2007), 57–76. ISSN: 0362-1340. DOI: 10.1145/1297105.1297033. URL: https://doi.org/10.1145/1297105.1297033.

[63] María Gómez, Romain Rouvoy, Martin Monperrus, and Lionel Seinturier. "A Recommender System of Buggy App Checkers for App Store Moderators." In: *MOBILESoft*. IEEE Computer Society, 2015, pp. 1–11.

[64] Google Help. *Share Files or Apps with Nearby Share*. Last accessed on Apr 2023. 2022. URL: https://support.google.com/files/answer/10514188.

[65] Google. *Android `LocationManager` documentation*. Last accessed on April 21st, 2024. 2011. URL: https://developer.android.com/reference/android/location/LocationManager#addTestProvider(java.lang.String,%20android.location.provider.ProviderProperties,%20java.util.Set%3Cjava.lang.String%3E).

[66] Google. *Android tcpdump*. Last accessed March 19th, 2026. 2015. URL: http://www.androidtcpdump.com/.

[67] Google. *Investigating Your RAM Usage*. Last accessed March 19th, 2026. 2016. URL: https://web.archive.org/web/20161107092506/https://developer.android.com/studio/profile/investigate-ram.html.

[68] Google. *App permissions best practices*. Last accessed March 23rd, 2026. 2018. URL: https://web.archive.org/web/20180716232359/https://developer.android.com/training/permissions/usage-notes.

[69] Google. *Flutter framework*. https://flutter.dev/. Last accessed on March 22nd, 2026. 2018.

[70] Google. *The Future of cloud computing*. Google, 2019.

[71] Google. *Advertising ID*. Last accessed March 20th, 2026. 2021. URL: https://support.google.com/googleplay/android-developer/answer/6048248?hl=en.

[72] Google. *Nearby Connections API*. Last accessed on Apr 2023. 2022. URL: https://developers.google.com/nearby/connections/.

[73] Google. *Wi-Fi Aware Overview | Android Documentation*. Last accessed on Apr 2023. 2023. URL: https://developer.android.com/guide/topics/connectivity/wifi-aware.

[74] Google. *Build for Billions*. Last accessed March 18th, 2026. 2024. URL: https://developer.android.com/docs/quality-guidelines/build-for-billions.

[75] Google. *Android (Go edition)*. Last accessed March 18th, 2026. 2026. URL: https://developer.android.com/guide/topics/androidgo.

[76] Google. *Manifest.permission*. Last accessed March 19th, 2026. 2026. URL: https://developer.android.com/reference/android/Manifest.permission.

[77] Alessandra Gorla, Ilaria Tavecchia, Florian Gross, and Andreas Zeller. "Checking App Behavior Against App Descriptions." In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 1025–1035. ISBN: 978-1-4503-2756-5. DOI: 10.1145/2568225.2568276. URL: http://doi.acm.org/10.1145/2568225.2568276.

[78] Florian Grützmacher, Benjamin Beichler, Albert Hein, Thomas Kirste, and Christian Haubelt. "Time and memory efficient online piecewise linear approximation of sensor signals." In: *Sensors* 18.6 (2018), p. 1672.

[79] Édouard Guégain, Rémy Raes, Noé Chachignot, Clément Quinton, and Romain Rouvoy. "AndroWatts: Unpacking the Power Consumption of Mobile Device's Components." In: *Proceedings of the 12th International Conference on Mobile Software Engineering and Systems (MOBILESoft'25)*. Ottawa, Canada, Apr. 2025. DOI: 10.1109/MOBILESoft66462.2025.00015. URL: https://hal.science/hal-04928609.

[80] Jiaping Gui, Stu Mcilroy, Mei Nagappan, and William G. J. Halfond. "Truth in Advertising: The Hidden Cost of Mobile Ads for Software Developers." In: *Proceedings of the 37th International Conference on Software Engineering (ICSE)*. May 2015.

[81] Yuelin Han, Zhifeng Wu, Pengfei Li, Adam Wierman, and Shaolei Ren. *The Unpaid Toll: Quantifying and Addressing the Public Health Impact of Data Centers*. 2025. arXiv: 2412.06288 [cs.CY]. URL: https://arxiv.org/abs/2412.06288.

[82] Ramaswamy Hariharan and Kentaro Toyama. "Project Lachesis: parsing and modeling location histories." In: *International Conference on Geographic Information Science*. Springer. 2004, pp. 106–124.

[83] Michael Haus, Muhammad Waqas, Aaron Yi Ding, Yong Li, Sasu Tarkoma, and Jörg Ott. "Security and Privacy in Device-to-Device (D2D) Communication: A Review." In: *IEEE Communications Surveys Tutorials* 19.2 (2017), pp. 1054–1079. ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2649687.

[84] Omar Hayat, Razali Ngah, Siti Zaiton Mohd Hashim, Muhammad Hashim Dahri, Reza Firsandaya Malik, and Yusnita Rahayu. "Device discovery in D2D communication: A survey." In: *IEEE Access* 7 (2019), pp. 131114–131134.

[85] Abram Hindle. "Green Software Engineering: The Curse of Methodology." In: *Leaders of Tomorrow Symposium: Future of Software Engineering, FOSE@SANER 2016, Osaka, Japan, March 14, 2016*. 2016, pp. 46–55. DOI: 10.1109/SANER.2016.60. URL: https://doi.org/10.1109/SANER.2016.60.

[86] Myles Hollander and Douglas A. Wolfe. *Nonparametric Statistical Methods*. New York: John Wiley & Sons, 1973.

[87] Jianjun Huang, Bo Xue, Jiasheng Jiang, Wei You, Bin Liang, Jingzheng Wu, and Yanjun Wu. "Scalably Detecting Third-Party Android Libraries With Two-Stage Bloom Filtering." In: *IEEE Transactions on Software Engineering* 49.4 (2023), pp. 2272–2284. DOI: 10.1109/TSE.2022.3215628.

[88] IEEE. *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. Standard. IEEE Std 802.11-1997, Nov. 1997. DOI: 10.1109/IEEESTD.1997.85951.

[89] IEEE. *IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High Speed Physical Layer in the 5 GHz Band*. Standard. IEEE Std 802.11a-1999, Dec. 1999. DOI: 10.1109/IEEESTD.1999.90606.

[90] IEEE. *IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher Speed Physical Layer (PHY) Extension in the 2.4 GHz Band*. Standard. IEEE Std 802.11b-1999, Jan. 2000. DOI: 10.1109/IEEESTD.2000.90914.

[91] IEEE. *IEEE Standard for Low-Rate Wireless Networks*. Standard. IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015), July 2020. DOI: 10.1109/IEEESTD.2020.9144691.

[92] IEEE. *IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN*. Standard. IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020), May 2021. DOI: 10.1109/IEEESTD.2021.9442429.

[93] IEEE. *IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Enhanced Throughput for Operation in License-Exempt Bands above 45 GHz*. Standard. IEEE Std 802.11ay-2021 (Amendment to IEEE Std 802.11-2020 as amendment by IEEE Std 802.11ax-2021), July 2021. DOI: 10.1109/IEEESTD.2021.9502046.

[94] IEEE. *IEEE P802.11 - TASK GROUP BH (RCM)*. Meeting Update. Last accessed on May 2023. 2022. URL: https://ieee802.org/11/Reports/tgbh_update.htm.

[95] ISO Standards. *ISO/IEC 18092:2004 — Information technology — Telecommunications and information exchange between systems — Near Field Communication — Interface and Protocol*. Standard. Apr. 2004.

[96] ISO Standards. *ISO/IEC 26907:2007 — Information Technology — Automatic Telecommunications and information exchange between systems — High Rate Ultra Wideband PHY and MAC Standard*. Standard. 2007.

[97] ISO Standards. *ISO/IEC 26908:2007 — Information Technology — Automatic Telecommunications and information exchange between systems — MAC-PHY Interface for ISO/IEC 26907*. Standard. 2007.

[98] ISO Standards. *ISO/IEC 18004:2015 — Information Technology — Automatic Identification and Data Capture Techniques — QR Code Bar Code Symbology Specification*. Standard. Feb. 2015.

[99] InfluxData. *InfluxDB*. https://www.influxdata.com/products/influxdb-overview/. Last accessed April 21st, 2024. 2013.

[100] Infrared Data Association. *Serial Infrared Physical Layer Specification*. Standard v1.4. 2001.

[101] Intergovernmental Panel on Climate Change. *Synthesis report: Climate Change*. Last accessed on February 16th, 2026. 2023. URL: https://www.ipcc.ch/report/ar6/syr/downloads/report/IPCC_AR6_SYR_FullVolume.pdf.

[102] Internet world stats. *INTERNET USAGE STATISTICS*. Last accessed on Apr 2023. 2021. URL: https://internetworldstats.com/stats.htm.

[103] Mohamed Sufyan Islim and Harald Haas. "Modulation Techniques for Li-Fi." In: *ZTE Communications* 14 (Apr. 2016).

[104] Shan Jaffry, Syed Faraz Hasan, Xiang Gui, and Yaw Wen Kuo. "Distributed device discovery in ProSe environments." In: *TENCON 2017-2017 IEEE Region 10 Conference*. IEEE. 2017, pp. 614–618.

[105] Furqan Jameel, Zara Hamid, Farhana Jabeen, Sherali Zeadally, and Muhammad Awais Javed. "A Survey of Device-to-Device Communications: Research Issues and Challenges." In: *IEEE Communications Surveys Tutorials* 20.3 (2018), pp. 2133–2168. ISSN: 1553-877X. DOI: 10.1109/COMST.2018.2828120.

[106] Tomasz Kalbarczyk and Christine Julien. "Omni: An Application Framework for Seamless Device-to-Device Interaction in the Wild." In: *Proceedings of the 19th International Middleware Conference on - Middleware '18*. Rennes, France: ACM Press, 2018, pp. 161–173. DOI: 10.1145/3274808.3274821.

[107] U. Kar and Debarshi Kumar Sanyal. "A Critical Review of 3GPP Standardization of Device-to-Device Communication in Cellular Networks." In: *SN Comput. Sci.* (2020). DOI: 10.1007/s42979-019-0045-5.

[108] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. "An online algorithm for segmenting time series." In: *Proceedings 2001 IEEE international conference on data mining*. IEEE. 2001, pp. 289–296.

[109] Besma Khalfoun, Sonia Ben Mokhtar, Sara Bouchenak, and Vlad Nitu. "EDEN: Enforcing Location Privacy through Re-identification Risk Assessment: A Federated Learning Approach." In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5.2 (2021). DOI: 10.1145/3463502.

[110] Martin Kleppmann, Adam Wiggins, Peter van Hardenberg, and Mark McGranaghan. "Local-first software: you own your data, in spite of the cloud." In: *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. Onward! 2019. New York, NY, USA: Association for Computing Machinery, 2019, 154–178. ISBN: 9781450369954. DOI: 10.1145/3359591.3359737. URL: https://doi.org/10.1145/3359591.3359737.

[111] Bran Knowles et al. "Climate Change: What is Computing's Responsibility? (Dagstuhl Perspectives Workshop 25122)." In: *Dagstuhl Manifestos* 11.1 (2025). Ed. by Bran Knowles et al., pp. 1–18. ISSN: 2193-2433. DOI: 10.4230/DagMan.11.1.1. URL: https://drops.dagstuhl.de/entities/document/10.4230/DagMan.11.1.1.

[112] Philip J Landrigan et al. "The Lancet Commission on pollution and health." In: *The Lancet* 391.10119 (2018), pp. 462–512. ISSN: 0140-6736. DOI: https://doi.org/10.1016/S0140-6736(17)32345-0. URL: https://www.sciencedirect.com/science/article/pii/S0140673617323450.

[113] Enrique Larios Vargas, Maurício Aniche, Christoph Treude, Magiel Bruntink, and Georgios Gousios. "Selecting third-party libraries: the practitioners' perspective." In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, 245–256. ISBN: 9781450370431. DOI: 10.1145/3368089.3409711. URL: https://doi.org/10.1145/3368089.3409711.

[114] Alexandru Lavric, Adrian I. Petrariu, and Valentin Popa. "SigFox Communication Protocol: The New Era of IoT?" In: *2019 International Conference on Sensing and Instrumentation in IoT Era (ISSI)*. Aug. 2019, pp. 1–4. DOI: 10.1109/ISSI47111.2019.9043727.

[115] Abner Li. *Pixel 6 Pro UWB, Adjust Assistant Long Press, Pixel Buds A-Series Bass Slider, & More*. Last accessed on Apr 2023. Dec. 2021. URL: https://9to5google.com/2021/12/06/december-pixel-feature-drop-android-12/.

[116] Menghao Li, Wei Wang, Pei Wang, Shuai Wang, Dinghao Wu, Jian Liu, Rui Xue, and Wei Huo. "LibD: scalable and precise third-party library detection in android markets." In: *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017*. 2017, pp. 335–346. DOI: 10.1109/ICSE.2017.38. URL: http://doi.ieeecomputersociety.org/10.1109/ICSE.2017.38.

[117] Pengfei Li, Jianyi Yang, Mohammad A. Islam, and Shaolei Ren. "Making AI Less 'Thirsty'." In: *Commun. ACM* 68.7 (June 2025), 54–61. ISSN: 0001-0782. DOI: 10.1145/3724499. URL: https://doi.org/10.1145/3724499.

[118] Mario Linares-Vásquez, Gabriele Bavota, Carlos Bernal-Cárdenas, Rocco Oliveto, Massimiliano Di Penta, and Denys Poshyvanyk. "Mining Energy-greedy API Usage Patterns in Android Apps: An Empirical Study." In: *Proceedings of the 11th Working Conference on Mining Software Repositories*. MSR 2014. New York, NY, USA: ACM, 2014, pp. 2–11. ISBN: 978-1-4503-2863-0. DOI: 10.1145/2597073.2597085. URL: http://doi.acm.org/10.1145/2597073.2597085.

[119] Yehuda Lindell. "Anonymous Authentication." In: *Journal of Privacy and Confidentiality* 2.2 (Apr. 2011). ISSN: 2575-8527. DOI: 10.29012/jpc.v2i2.590.

[120] Xiaoyan Liu, Zhenjiang Lin, and Huaiqing Wang. "Novel online methods for time series segmentation." In: *IEEE Transactions on Knowledge and Data Engineering* 20.12 (2008), pp. 1616–1626.

[121] Bingqing Luo, Jincheng Gao, and Zhixin Sun. "Energy Modeling of Neighbor Discovery in Bluetooth Low Energy Networks." In: *Sensors* 19.22 (Jan. 2019). ISSN: 1424-8220. DOI: 10.3390/s19224997. (Visited on 03/06/2023).

[122] Shutian Luo, Huanle Xu, Chengzhi Lu, Kejiang Ye, Guoyao Xu, Liping Zhang, Yu Ding, Jian He, and Chengzhong Xu. "Characterizing Microservice Dependency and Performance: Alibaba Trace Analysis." In: *Proceedings of the ACM Symposium on Cloud Computing*. 2021, pp. 412–426. URL: https://github.com/alibaba/clusterdata/blob/master/cluster-trace-microservices-v2021.

[123] Adrien Luxey. "E-Squads: A Novel Paradigm to Build Privacy-Preserving Ubiquitous Applications." PhD thesis. Université de Rennes, Nov. 2019, pp. 7–8,12.

[124] Adrien Luxey, Yérom-David Bromberg, Fábio M. Costa, Vinícius Lima, Ricardo C. A. da Rocha, and François Taïani. "Sprinkler: A probabilistic dissemination protocol to provide fluid user interaction in multi-device ecosystems." In: *IEEE Inter-*

national Conference on Pervasive Computing and Communications (PerCom). 2018, pp. 1–10.

[125] Yingjun Lyu, Jiaping Gui, Mian Wan, and William G. J. Halfond. "An Empirical Study of Local Database Usage in Android Applications." In: *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME)*. Sept. 2017.

[126] Ziang Ma, Haoyu Wang, Yao Guo, and Xiangqun Chen. "LibRadar: Fast and Accurate Detection of Third-party Libraries in Android Apps." In: *Proceedings of the 38th International Conference on Software Engineering Companion*. ICSE '16. New York, NY, USA: ACM, 2016, pp. 653–656. ISBN: 978-1-4503-4205-6. DOI: 10.1145/2889160.2889178. URL: http://doi.acm.org/10.1145/2889160.2889178.

[127] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. "l-Diversity: Privacy beyond k-anonymity." In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), 3–es.

[128] Erick Macias and Josh Wyatt. *NFC Active and Passive Peer-to-Peer Communication Using the TRF7970A*. Tech. rep. SLOA192B. Texas Instruments, Apr. 2014, p. 33.

[129] Kent Madsen. *Hardware Performance Archive [2019]*. 2022. DOI: 10.34740/KAGGLE/DS/2408839. URL: https://www.kaggle.com/ds/2408839.

[130] Mohamed Maouche, Sonia Ben Mokhtar, and Sara Bouchenak. "HMC: Robust privacy protection of mobility data against multiple re-identification attacks." In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2.3 (2018), pp. 1–25.

[131] Mohamed Maouche, Sonia Ben Mokhtar, and Sara Bouchenak. "Ap-attack: a novel user re-identification attack on mobility datasets." In: *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. 2017, pp. 48–57.

[132] Raquel A. Martins, Ivo Sousa, Maria Paula Queluz, and António Rodrigues. "Interference Cancellation Techniques for Device-to-Device Discovery in Out-of-Coverage Networks." In: *IEEE Access* 9 (2021). ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3050256.

[133] S. C. Matz, M. Kosinski, G. Nave, and D. J. Stillwell. "Psychological Targeting as an Effective Approach to Digital Mass Persuasion." In: *Proceedings of the National Academy of Sciences* 114.48 (Nov. 2017). ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1710966114.

[134] Colleen Mcclain. "34% of Lower-Income Home Broadband Users Have Had Trouble Paying for Their Service amid COVID-19." In: *Pew Research Center* (June 2021). (Visited on 05/15/2023).

[135] Lakhdar Meftah, Romain Rouvoy, and Isabelle Chrisment. "FOUGERE: user-centric location privacy in mobile crowd-sourcing apps." In: *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer. 2019, pp. 116–132.

[136] Lakhdar Meftah, Romain Rouvoy, and Isabelle Chrisment. "Testing Nearby Peer-to-Peer Mobile Apps at Large." In: *MOBILESoft 2019 - 6th IEEE/ACM International Conference on Mobile Software Engineering and Systems*. May 2019. DOI: 10.1109/MOBILESoft.2019.00009.

[137] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. "Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT." In: *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. Mar. 2018, pp. 197–202. DOI: 10.1109/PERCOMW.2018.8480255.

[138] Microsoft. *Environmental Sustainability Report*. Tech. rep. Microsoft, 2025.

[139] Nimantha Mihirange. *Why Are iPhones Unable to Transfer Files to Android Phones via Bluetooth?* Jan. 2017. URL: https://www.quora.com/Why-are-iPhones-unable-to-transfer-files-to-Android-phones-via-Bluetooth (visited on 03/14/2023).

[140] Roberto Minelli and Michele Lanza. "Software Analytics for Mobile Applications-Insights & Lessons Learned." In: *17th European Conference on Software Maintenance and Reengineering, CSMR 2013, Genova, Italy, March 5-8, 2013*. 2013, pp. 144–153. DOI: 10.1109/CSMR.2013.24. URL: https://doi.org/10.1109/CSMR.2013.24.

[141] Vikas Mishra, Pierre Laperdrix, Antoine Vastel, Walter Rudametkin, Romain Rouvoy, and Martin Lopatka. "Don't Count Me Out: On the Relevance of IP Address In The Tracking Ecosystem." In: *Proceedings of The Web Conference 2020*. WWW '20. New York, NY, USA: ACM, 2020. ISBN: 9781450370233. DOI: 10.1145/3366423.3380161.

[142] Assaad Moawad, Thomas Hartmann, François Fouquet, Grégory Nain, Jacques Klein, and Yves Le Traon. "Beyond discrete modeling: A continuous and efficient model for IoT." In: *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE. 2015, pp. 90–99.

[143] Jayashree Mohan, Dhathri Purohith, Matthew Halpern, Vijay Chidambaram, and Vijay Janapa Reddi. "Storage on Your SmartPhone Uses More Energy Than You Think." In: *9th USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage 2017, Santa Clara, CA, USA, July 10-11, 2017*. 2017. URL: https://www.usenix.org/conference/hotstorage17/program/presentation/mohan.

[144] Sonia Ben Mokhtar, Antoine Boutet, Louafi Bouzouina, Patrick Bonnel, Olivier Brette, Lionel Brunie, Mathieu Cunche, Stephane D'Alu, Vincent Primault, Patrice Raveneau, et al. "PRIVA'MOV: Analysing Human Mobility Through Multi-Sensor Datasets." In: *NetMob 2017*. 2017.

[145] Emilio Molina, Mirko Fiacchini, Sophie Cerf, and Bogdan Robu. "Optimal privacy protection of mobility data: a predictive approach." In: *IFAC-PapersOnLine* 56.2 (2023). 22nd IFAC World Congress, pp. 11015–11020. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2023.10.801. URL: https://www.sciencedirect.com/science/article/pii/S2405896323011783.

[146] Emilio Molina, Mirko Fiacchini, Arthur Goarant, Rémy Raes, Sophie Cerf, and Bogdan Robu. "Application of a predictive method to protect privacy of mobility data." In: *Control Engineering Practice* 156 (Mar. 2025), p. 106223. DOI: 10.1016/j.conengprac.2024.106223. URL: https://hal.science/hal-04885266.

[147] Emilio Molina, Mirko Fiacchini, Arthur Goarant, Rémy Raes, Sophie Cerf, and Bogdan Robu. "Application of a predictive method to protect privacy of mobility data." In: *Control Engineering Practice* 156 (2025), p. 106223. ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2024.106223. URL: https://www.sciencedirect.com/science/article/pii/S0967066124003824.

[148] Monsoon Solutions, Inc. *Power Monitor Software*. Last accessed March 19th, 2026. 2015. URL: https://msoon.github.io/powermonitor/.

[149] Rodrigo Morales, Rubén Saborido, Foutse Khomh, Francisco Chicano, and Giuliano Antoniol. "EARMO: An Energy-Aware Refactoring Approach for Mobile Apps." In: *IEEE Transactions on Software Engineering* 44.12 (2018), pp. 1176–1206. DOI: 10.1109/TSE.2017.2757486.

[150] Léa Mosesso, Nolwenn Maudet, Edlira Nano, Thomas Thibault, and Aurélien Tabard. "Obsolescence Paths: living with aging devices." In: *ICT4S 2023 - International Conference on Information and Communications Technology for Sustainability*. Rennes, France,

June 2023. DOI: 10.1109/ICT4S58814.2023.00011. URL: https://hal.science/hal-04097867.

[151] San Murugesan. "Harnessing Green IT: Principles and Practices." In: *IT Professional* 10.1 (2008), pp. 24–33. DOI: 10.1109/MITP.2008.10.

[152] NASA. *Carbon Dioxide - Earth Indicator*. Last accessed on April 28th, 2026. 2026. URL: https://science.nasa.gov/earth/explore/earth-indicators/carbon-dioxide/.

[153] Aamir Nadeem and Ho-Shin Cho. "Genetic Algorithm-Based Energy Efficiency Maximization for Social-Aware Device-to-Device Communications." In: *IEEE Access* 9 (2021). ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3079108.

[154] Ellen Nakashima and Joby Warrick. "For NSA Chief, Terrorist Threat Drives Passion to 'Collect It All'." In: *Washington Post* (July 2013). ISSN: 0190-8286.

[155] National Renewable Energy Laboratory. *Solar Power Data for Integration Studies*. 2006. URL: https://www.nrel.gov/grid/solar-power-data.

[156] Richard Nieva. *Google's Pixel 6 Phones Are Coming with a Chip Designed in-House*. Last accessed on Apr 2023. Aug. 2021. URL: https://www.cnet.com/tech/mobile/googles-pixel-6-phones-are-coming-with-a-chip-the-apple-rival-designed-in-house/.

[157] Nintendo Wiki. *List of Nintendo Systems*. Last accessed on Apr 2023. 2022. URL: https://nintendo.fandom.com/wiki/List_of_Nintendo_systems.

[158] Travis Oliphant. *NumPy website*. Last accessed on March 23rd, 2026. 2006. URL: https://numpy.org/.

[159] Federica Paci, Jacopo Pizzoli, and Nicola Zannone. "A Comprehensive Study on Third-Party User Tracking in Mobile Applications." In: *Proceedings of the 18th International Conference on Availability, Reliability and Security*. ARES '23. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9798400707728. DOI: 10.1145/3600160.3605079. URL: https://doi.org/10.1145/3600160.3605079.

[160] Farzaneh Pakzad, Marius Portmann, Thierry Turletti, Thierry Parmentelat, Mohamed Naoufal Mahfoudi, and Walid Dabbous. "R2Lab Testbed Evaluation for Wireless Mesh Network Experiments." In: *ITNAC*. IEEE Computer Society, 2018, pp. 1–6.

[161] Rahul Pandita, Xusheng Xiao, Wei Yang, William Enck, and Tao Xie. "WHYPER: Towards Automating Risk Assessment of Mobile Applications." In: *Proceedings of the 22Nd USENIX Conference on Security*. SEC'13. Washington, D.C.: USENIX Association, 2013, pp. 527–542. ISBN: 978-1-931971-03-4. URL: http://dl.acm.org/citation.cfm?id=2534766.2534812.

[162] David Patterson, Jeffrey M. Gilbert, Marco Gruteser, Efren Robles, Krishna Sekar, Yong Wei, and Tenghui Zhu. "Energy and Emissions of Machine Learning on Smartphones vs. the Cloud." In: *Commun. ACM* 67.2 (Jan. 2024), 86–97. ISSN: 0001-0782. DOI: 10.1145/3624719. URL: https://doi.org/10.1145/3624719.

[163] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. "CRAWDAD EPFL/MOBILITY." In: *IEEE Dataport*. 2022. DOI: 10.15783/C7J010. URL: https://dx.doi.org/10.15783/C7J010.

[164] Hans-Juergen Pirch and Frank Leong. *Introduction to Impulse Radio UWB Seamless Access Systems*. White paper. FiRa Consortium, 2020, p. 15.

[165] Polar. *Ignite 2*. https://www.polar.com/en/ignite2. Last accessed on April 21st, 2024. 2021.

[166] Politico. *Google makes changes to privacy oversight, worrying policymakers*. Last accessed on December 5th, 2025. 2024. URL: https://www.politico.com/news/2024/07/11/google-privacy-oversight-changes-worry-congress-00167427.

[167] Rob Price. "Google Drive Now Hosts More than 2 Trillion Files." In: *Business Insider* (May 2017). (Visited on 04/19/2023).

[168] Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, and Lionel Brunie. "The Long Road to Computational Location Privacy: A Survey." In: *IEEE Communications Surveys Tutorials* 21.3 (2019), pp. 2772–2793. ISSN: 1553-877X. DOI: 10.1109/COMST.2018.2873950.

[169] Vincent Primault, Sonia Ben Mokhtar, Cédric Lauradoux, and Lionel Brunie. "Differentially private location privacy in practice." In: *arXiv preprint arXiv:1410.7744* (2014).

[170] Vincent Primault, Sonia Ben Mokhtar, Cédric Lauradoux, and Lionel Brunie. "Time distortion anonymization for the publication of mobility data with high utility." In: *2015 IEEE Trustcom/BigDataSE/ISPA*. Vol. 1. IEEE. 2015, pp. 539–546.

[171] Zhao Q, Guo Y, Ye T, and al. "Global, regional, and national burden of mortality associated with non-optimal ambient temperatures from 2000 to 2019: a three-stage modelling study." In: *Lancet Planet Health* (2021). DOI: 10.1016/S2542-5196(21)

00081-4. URL: https://www.thelancet.com/journals/lanplh/article/PIIS2542-5196(21)00081-4.

[172] Rémy Raes, Olivier Ruas, Adrien Luxey-Bitri, and Romain Rouvoy. *FLI implementation*. Hosted on Software Heritage. Last accessed on November 4th, 2024. 2022.

[173] Rémy Raes, Olivier Ruas, Adrien Luxey-Bitri, and Romain Rouvoy. *In-situ LPPM*. Hosted on Software Heritage. Last accessed on November 4th, 2024. 2022.

[174] Rémy Raes, Olivier Ruas, Adrien Luxey-Bitri, and Romain Rouvoy. *Memory space benchmarking application*. Hosted on Software Heritage. Last accessed on November 4th, 2024. 2022.

[175] Rémy Raes, Olivier Ruas, Adrien Luxey-Bitri, and Romain Rouvoy. *Throughput benchmarking application*. Hosted on Software Heritage. Last accessed on November 4th, 2024. 2022.

[176] Rémy Raes, Olivier Ruas, Adrien Luxey-Bitri, and Romain Rouvoy. *Fast Linear Interpolation (FLI) accelerometer example application*. Hosted on Software Heritage. Last accessed on November 4th, 2024. 2022.

[177] Rémy Raes. *FLInD notebook*. Last accessed on March 17th, 2026. 2026. URL: https://archive.softwareheritage.org/browse/origin/directory/?origin_url=https://github.com/Alystrasz/less-is-more.

[178] Rémy Raes, Adrien Luxey-Bitri, and Daniel Romero Acero. *Venice source code repository*. Last accessed on March 23rd, 2026. 2023. URL: https://github.com/Venice-D2D.

[179] Rémy Raes, Adrien Luxey-Bitri, Romain Rouvoy, Davide Frey, and François Taïani. "Venice: eschewing the cloud by leveraging local communication channels." In: *Joint Proceedings of ICT4S 2024 Doctoral Symposium, Demonstrations & Posters Track and Workshops*. Stockholm, Sweden, June 2024, pp. 1–4. URL: https://hal.science/hal-04576743.

[180] Rémy Raes, Olivier Ruas, Adrien Luxey-Bitri, and Romain Rouvoy. "Compact Storage of Data Streams in Mobile Devices." In: *Proceedings of the 24th International Conference on Distributed Applications and Interoperable Systems (DAIS'24)*. LNCS. Groningen, Netherlands: Springer, June 2024. DOI: 10.1007/978-3-031-62638-8_4. URL: https://hal.science/hal-04535716.

[181] Rémy Raes, Olivier Ruas, Adrien Luxey-Bitri, and Romain Rouvoy. "INTACT: Compact Storage of Data Streams in Mobile Devices to Unlock User Privacy at the Edge." In: *Journal of Internet Services and Applications (JISA)* 16.1 (June 2025), 372–387. DOI: 10.5753/jisa.2025.5242. URL: https://journals-sol.sbc.org.br/index.php/jisa/article/view/5242.

[182] Michael Rowan-Robinson. *Night Vision: Exploring the Infrared Universe*. Cambridge: Cambridge University Press, 2012, p. 23. ISBN: 978-1-107-02476-2.

[183] Israel J. Mojica Ruiz, Meiyappan Nagappan, Bram Adams, Thorsten Berger, Steffen Dienst, and Ahmed E. Hassan. "Impact of Ad Libraries on Ratings of Android Mobile Apps." In: *IEEE Software* 31.6 (2014), pp. 86–92. ISSN: 0740-7459. DOI: doi.ieeecomputersociety.org/10.1109/MS.2014.79.

[184] Rubén Saborido, Rémy Raes, Rodrigo Morales, Romain Rouvoy, Foutse Khomh, and Yann-Gaël Guéhéneuc. "Simply the Best -A Systematic Evaluation Approach for Third-Party Libraries based on Mobile App Quality Attributes." In: *Empirical Software Engineering* 31.5 (May 2026), p. 137. DOI: 10.1007/s10664-026-10872-w. URL: https://link.springer.com/article/10.1007/s10664-026-10872-w.

[185] Rubén Saborido, Foutse Khomh, Giuliano Antoniol, and Yann-Gaël Guéhéneuc. "Comprehension of Ads-supported and Paid Android Applications: Are They Different?" In: *Proceedings of the 25th International Conference on Program Comprehension (ICPC)*. Buenos Aires, Argentina: IEEE, May 2017, pp. 143–153. DOI: 10.1109/ICPC.2017.25.

[186] Rubén Saborido, Rodrigo Morales, Foutse Khomh, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. "Getting the most from map data structures in Android." In: *Empirical Software Engineering* 23.5 (Oct. 2018), pp. 2829–2864. ISSN: 1573-7616. DOI: 10.1007/s10664-018-9607-8. URL: https://doi.org/10.1007/s10664-018-9607-8.

[187] Cagri Sahin, Philip Tornquist, Ryan Mckenna, Zachary Pearson, and James Clause. "How Does Code Obfuscation Impact Energy Usage?" In: *ICSME'14*. 2014, pp. 131–140.

[188] K.V.S.S.S.S. Sairam, N. Gunasekaran, and S.R. Redd. "Bluetooth in Wireless Communication." In: *IEEE Communications Magazine* 40.6 (June 2002), pp. 90–96. ISSN: 1558-1896. DOI: 10.1109/MCOM.2002.1007414.

[189] H. Sakoe and S. Chiba. "Dynamic programming algorithm optimization for spoken word recognition." In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1 (1978), pp. 43–49. DOI: 10.1109/TASSP.1978.1163055. URL: https://ieeexplore.ieee.org/document/1163055.

[190] Schwayne. *iOS and Wi-Fi Direct | Apple Developer Forums*. 2018. URL: https://developer.apple.com/forums/thread/12885?page=2 (visited on 03/14/2023).

[191] Secure Mobile Networking Lab. *OpenDrop: An Open Source AirDrop Implementation*. Last accessed in Apr 2023. Aug. 2022. URL: https://github.com/seemoo-lab/opendrop.

[192] Wenlong Shen, Weisheng Hong, Xianghui Cao, Bo Yin, Devu Manikantan Shila, and Yu Cheng. "Secure Key Establishment for Device-to-Device Communications." In: *2014 IEEE Global Communications Conference*. Dec. 2014, pp. 336–340. DOI: 10.1109/GLOCOM.2014.7036830.

[193] Yi-Sheng Shiu, Shih Chang, Hsiao-Chun Wu, Scott Huang, and Hsiao-Hwa Chen. "Physical Layer Security in Wireless Networks: A Tutorial." In: *IEEE Wireless Communications* 18.2 (Apr. 2011), pp. 66–74. ISSN: 1536-1284. DOI: 10.1109/MWC.2011.5751298.

[194] Richard Speed. *Builder.ai coded itself into a corner – now it's bankrupt*. Last accessed on December 5th, 2025. 2025. URL: https://www.theregister.com/2025/05/21/builderai_insolvency/.

[195] Milan Stute, David Kreitschmann, and Matthias Hollick. "One Billion Apples' Secret Sauce: Recipe for the Apple Wireless Direct Link Ad Hoc Protocol." In: *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (Oct. 2018), pp. 529–543. DOI: 10.1145/3241539.3241566. eprint: 1808.03156.

[196] Milan Stute, David Kreitschmann, and Matthias Hollick. *The Open Wireless Link Project*. 2018. URL: https://owlink.org.

[197] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. "A Billion Open Interfaces for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link." In: *28th USENIX Security Symposium (USENIX Security 19)*. 2019, pp. 37–54. ISBN: 978-1-939133-06-9.

[198] Aurore Stéphant. *Mining rush in the 21st century: how far will the limits be pushed? (French)*. Last accessed on February 9th, 2026. 2022. URL: https://www.youtube.com/watch?v=i8RMX8ODWQs.

[199] Wanlu Sun, Mohammad Reza Gholami, Erik G Ström, and Fredrik Brännström. "Distributed clock synchronization with application of D2D communication without infrastructure." In: *2013 IEEE Globecom Workshops (GC Wkshps)*. IEEE. 2013, pp. 561–566.

[200] Latanya Sweeney. "k-anonymity: A model for protecting privacy." In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 557–570.

[201] Tails. *MAC Address Anonymization*. Documentation. Last accessed on May 2023. 2022. URL: https://tails.boum.org/contribute/design/MAC_address/.

[202] James Tamplin and Andrew Lee. *Firebase services*. https://firebase.google.com. Last accessed on March 21st, 2026. 2012.

[203] The Android Open Source Project. *Android Input.java class*. Last accessed March 22nd, 2026. 2007. URL: https://android.googlesource.com/platform/frameworks/base/+/android-4.4.2_r1/cmds/input/src/com/android/commands/input/Input.java\#271.

[204] Timescale Inc. *Timescale database*. https://www.timescale.com. Last accessed on April 21st, 2024. 2018.

[205] Timescale. *Building a Distributed Time-Series Database on PostgreSQL*. Last accessed on April 29th 2026. Aug. 2019. URL: https://web.archive.org/web/20220423172804/https://www.timescale.com/blog/building-a-distributed-time-series-database-on-postgresql/.

[206] C.-K. Toh, Richard Chen, Minar Delwar, and Donald Allen. "Experimenting with an Ad Hoc Wireless Network on Campus: Insights and Experiences." In: *ACM SIGMETRICS Performance Evaluation Review* 28.3 (Dec. 2000), pp. 21–29. ISSN: 0163-5999. DOI: 10.1145/377616.377622.

[207] C.-K. Toh, M. Delwar, and D. Allen. "Evaluating the Communication Performance of an Ad Hoc Wireless Network." In: *IEEE Transactions on Wireless Communications* 1.3 (July 2002), pp. 402–414. ISSN: 1558-2248. DOI: 10.1109/TWC.2002.800539.

[208] Chai Keong Toh. "Routing Method for Ad-Hoc Mobile Networks." Pat. US5987011A. Nov. 1999.

[209] Sacha Trifunovic, Sylvia T. Kouyoumdjieva, Bernhard Distl, Ljubica Pajevic, Gunnar Karlsson, and Bernhard Plattner. "A Decade of Research in Opportunistic Networks: Challenges, Relevance, and Future Directions." In: *IEEE Communications Magazine* 55.1 (Jan. 2017), pp. 168–173. ISSN: 1558-1896. DOI: 10.1109/MCOM.2017.1500527CM.

[210] Artur Trindade. *ElectricityLoadDiagrams20112014*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C58C86. 2015.

[211] United Nations. *With Almost Half of World's Population Still Offline, Digital Divide Risks Becoming 'New Face of Inequality', Deputy Secretary-General Warns General Assembly*. Last accessed on Apr 2023. 2021. URL: https://www.un.org/press/en/2021/dsgsm1579.doc.htm.

[212] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. "Recognizing Detailed Human Context in the Wild from Smartphones and Smartwatches." In: *IEEE Pervasive Computing* 16.4 (Oct. 2017). ISSN: 1558-2590. DOI: 10.1109/MPRV.2017.3971131.

[213] Stefan Viehböck. "Brute Forcing Wi-Fi Protected Setup." In: (Dec. 2011). Last accessed on May 2023. URL: https://www.cs.cmu.edu/~rdriley/330/papers/viehboeck_wps.pdf.

[214] T. K. Vintsyuk. "Speech discrimination by dynamic programming." In: *Cybernetics* 4 (1968), pp. 52–57. URL: https://api.semanticscholar.org/CorpusID:123081024.

[215] Emily A Vogels. "Some digital divides persist between rural, urban and suburban America." In: *Pew Research Center, August* 19 (2021).

[216] Haoyu Wang, Yao Guo, Ziang Ma, and Xiangqun Chen. "WuKong: A Scalable and Accurate Two-phase Approach to Android App Clone Detection." In: *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. ISSTA 2015. New York, NY, USA: ACM, 2015, pp. 71–82. ISBN: 978-1-4503-3620-8. DOI: 10.1145/2771783.2771795. URL: http://doi.acm.org/10.1145/2771783.2771795.

[217] Lin Wang, Hristijan Gjoreski, Mathias Ciliberto, Sami Mekki, Stefan Valentin, and Daniel Roggen. "Enabling reproducible research in sensor-based transportation mode recognition with the Sussex-Huawei dataset." In: *IEEE Access* 7 (2019), pp. 10870–10891.

[218] Mingjun Wang and Zheng Yan. "A Survey on Security in D2D Communications." In: *Mobile Networks and Applications* 22.2 (Apr. 2017), pp. 195–208. ISSN: 1572-8153. DOI: 10.1007/s11036-016-0741-5.

[219] Roy Want. "An introduction to RFID technology." In: *IEEE Pervasive Computing* 5 (Feb. 2006). DOI: 10.1109/MPRV.2006.2.

[220] Stephen A. Weis. "RFID (Radio Frequency IDentification): Principles and Applications." In: 2007.

[221] Wi-Fi Alliance. *Discover Wi-Fi Direct*. Last accessed on Apr 2023. 2023. URL: https://www.wi-fi.org/discover-wi-fi/wi-fi-direct.

[222] Wi-Fi Alliance. *Discover Wi-Fi Easy Connect*. Last accessed on Apr 2023. 2023. URL: https://www.wi-fi.org/discover-wi-fi/wi-fi-easy-connect.

[223] Wi-Fi Alliance. *Wi-Fi Aware*. Last accessed on Apr 2023. 2023. URL: https://www.wi-fi.org/discover-wi-fi/wi-fi-aware.

[224] Wikipedia. *List of Bluetooth Profiles*. Last accessed on Apr 2023. 2022. URL: https://en.wikipedia.org/wiki/List_of_Bluetooth_profiles.

[225] Wikipedia. *Internet Traffic*. Last accessed on May 2023. 2023. URL: https://en.wikipedia.org/wiki/Internet_traffic.

[226] O. Wolfson, S. Chamberlain, Son Dao, Liqin Jiang, and G. Mendez. "Cost and imprecision in modeling the position of moving objects." In: *Proceedings 14th International Conference on Data Engineering*. 1998, pp. 588–596. DOI: 10.1109/ICDE.1998.655822.

[227] World Health Organization. *Air quality, energy and health*. Last accessed on April 28th, 2026. 2025. URL: https://www.who.int/teams/environment-climate-change-and-health/air-quality-energy-and-health/health-impacts.

[228] World Wide Fund for Nature. *Living Planet Report*. Tech. rep. Last accessed on April 28th, 2026. WWF, 2024. URL: https://www.worldwildlife.org/publications/2024-living-planet-report/.

[229] Rest of World. *The life and death of SNET, Havana's alternative internet*. Last accessed on April 27th, 2026. 2020. URL: https://restofworld.org/2020/the-life-and-death-of-snet-havanas-alternative-internet/.

[230] Xiaomi Inc. *Mi Remote controller - for TV, STB, AC and more – Apps on Google Play*. Last accessed on Apr 2023. 2022. URL: https://play.google.com/store/apps/details?id=com.duokan.phone.remotecontroller.

[231] Daniel Yergin, Carlos Pascual, Eleonor Kramarz, Aurian De La Noue, Miguel Acosta, Gray Bender, Mohsen Bonakdarpour, Tabitha Bailey, and Keerti Rajan. *Copper in the Age of AI: Challenges of Electrification*. Last accessed on January 26th, 2026. 2026. URL: https://www.spglobal.com/en/research-insights/special-reports/copper-in-the-age-of-ai.

[232] Meng-Chieh Yu, Tong Yu, Shao-Chen Wang, Chih-Jen Lin, and Edward Y Chang. "Big data small footprint: The design of a low-power classifier for detecting transportation modes." In: *Proceedings of the VLDB Endowment* 7.13 (2014), pp. 1429–1440.

[233] Xian Zhan, Tianming Liu, Lingling Fan, Li Li, Sen Chen, Xiapu Luo, and Yang Liu. "Research on Third-Party Libraries in Android Apps: A Taxonomy and Systematic Literature Review." In: *IEEE Transactions on Software Engineering* 48.10 (2022), pp. 4181–4213. DOI: 10.1109/TSE.2021.3114381.

[234] Changqing Zhou, Dan Frankowski, Pamela Ludford, Shashi Shekhar, and Loren Terveen. "Discovering personal gazetteers: an interactive clustering approach." In: *Proceedings of the 12th annual ACM international workshop on Geographic information systems*. 2004, pp. 266–273.

[235] Zhenyu Zhou, Mianxiong Dong, Kaoru Ota, Jun Wu, and Takuro Sato. "Energy efficiency and spectral efficiency tradeoff in device-to-device (D2D) communications." In: *IEEE Wireless Communications Letters* 3.5 (2014), pp. 485–488.

[236] Shoshana Zuboff. *The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power*. New York: PublicAffairs, 2019. ISBN: 978-1-61039-569-4.

[237] Francisco Gomes de Oliveira Neto, Richard Torkar, Robert Feldt, Lucas Gren, Carlo A. Furia, and Ziwei Huang. "Evolution of statistical analysis in empirical software engineering research: Current state and steps forward." In: *Journal of Systems and Software* 156 (2019), pp. 246–267. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2019.07.002. URL: https://www.sciencedirect.com/science/article/pii/S0164121219301451.

Part I

APPENDIX

A

SERIES CLUSTERING FOR FLIND

One model for several series	One model per series	Size (floats) ↓
CPU#1, CPU#2, CPU#3, CPU#4	GPU Memory	385,356
CPU#1, CPU#2, CPU#3	CPU#4, GPU Memory	462,164
CPU#1, CPU#2, CPU#4	CPU#3, GPU Memory	462,218
CPU#1, CPU#3, CPU#4	CPU#2, GPU Memory	462,222
CPU#2, CPU#3, CPU#4	CPU#1, GPU Memory	462,306
CPU#1, CPU#2, CPU#3, CPU#4, GPU Memory	∅	462,408
CPU#1, CPU#2	CPU#3, CPU#4, GPU Memory	539,024
CPU#1, CPU#3	CPU#2, CPU#4, GPU Memory	539,030
CPU#1, CPU#4	CPU#2, CPU#3, GPU Memory	539,083
CPU#2, CPU#3	CPU#1, CPU#4, GPU Memory	539,118
CPU#2, CPU#4	CPU#1, CPU#3, GPU Memory	539,168
CPU#3, CPU#4	CPU#1, CPU#2, GPU Memory	539,174
CPU#1, CPU#2, CPU#3, GPU Memory	CPU#4	539,212
CPU#1, CPU#2, CPU#4, GPU Memory	CPU#3	539,270
CPU#1, CPU#3, CPU#4, GPU Memory	CPU#2	539,272
CPU#2, CPU#3, CPU#4, GPU Memory	CPU#1	539,358
CPU#1	CPU#2, CPU#3, CPU#4, GPU Memory	615,796
CPU#2	CPU#1, CPU#3, CPU#4, GPU Memory	615,796
CPU#3	CPU#1, CPU#2, CPU#4, GPU Memory	615,796
GPU Memory	CPU#1, CPU#2, CPU#3, CPU#4	615,796
CPU#4	CPU#1, CPU#2, CPU#3, GPU Memory	615,798
∅	CPU#1, CPU#2, CPU#3, CPU#4, GPU Memory	615,798
CPU#1, CPU#2, GPU Memory	CPU#3, CPU#4	616,070
CPU#1, CPU#3, GPU Memory	CPU#2, CPU#4	616,074
CPU#1, CPU#4, GPU Memory	CPU#2, CPU#3	616,132
CPU#2, CPU#3, GPU Memory	CPU#1, CPU#4	616,166
CPU#2, CPU#4, GPU Memory	CPU#1, CPU#3	616,220
CPU#3, CPU#4, GPU Memory	CPU#1, CPU#2	616,224
CPU#4, GPU Memory	CPU#1, CPU#2, CPU#3	692,728
CPU#3, GPU Memory	CPU#1, CPU#2, CPU#4	692,744
CPU#2, GPU Memory	CPU#1, CPU#3, CPU#4	692,750
CPU#1, GPU Memory	CPU#2, CPU#3, CPU#4	692,788

Table A.1: Results of computing every modelling combination of splitting series between one FLInD instance and one FLInD instance per model, sorted by ascending size. The most optimal way to store the five series is to store all CPU series within one FLInD instance, and model the *GPU Memory* series separately.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Final Version as of August 6, 2026 (classicthesis v4.8).