Privacy-preserving distributed queries compatible with opportunistic networks

Ludovic Javet

université PARIS-SACLAY

Privacy-preserving distributed queries compatible with opportunistic networks

Requêtes distribuées respectueuses de la vie privée compatibles avec des réseaux opportunistes

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 : Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique
Graduate School : Informatique et Sciences du Numérique
Référent : Université de Versailles-Saint-Quentin-en-Yvelines

Thèse préparée dans l'unité de recherche **Données et Algorithmes pour une ville intelligente et durable** (Université Paris-Saclay, UVSQ, Inria), sous la direction de **Luc BOUGANIM**, Directeur de recherche, le co-encadrement de **Nicolas ANCIAUX**, Directeur de recherche, et le co-encadrement de **Philippe PUCHERAL**, Professeur.

Thèse soutenue à Versailles, le 19 juillet 2023, par

Ludovic JAVET

Composition du Jury

Membres du jury avec voix délibérative

Nathalie MITTON Directrice de recherche, Inria	Présidente
Philippe LAMARRE Professeur, INSA Lyon	Rapporteur & Examinateur
Pierre SENS Professeur, Sorbonne Université	Rapporteur & Examinateur
Zoubida KEDAD Professeure, UVSQ, Université Paris-Saclay	Examinatrice
Claudia RONCANCIO Professeure, Grenoble INP Ensimag	Examinatrice

THESE DE DOCTORAT

NNT : 2023UPASG038

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Requêtes distribuées respectueuses de la vie privée compatibles avec des réseaux opportunistes

Mots clés : Gestion de données décentralisée, Calculs par la foule, Protection de la vie privée, Réseaux opportunistes

Résumé : Dans la société actuelle, où l'IoT et les plateformes numériques transforment notre vie quotidienne, les données personnelles sont générées à profusion et leur utilisation échappe souvent à notre contrôle. Des législations récentes comme le RGPD en Europe proposent des solutions concrètes pour réguler ces nouvelles pratiques et protéger notre vie privée. Parallèlement, sur le plan technique, de nouvelles architectures émergent pour répondre à ce besoin urgent de se réapproprier nos propres données personnelles. C'est le cas des systèmes de gestion des données personnelles (PDMS) qui offrent un moyen décentralisé de stocker et de gérer les données personnelles, permettant aux individus d'avoir un meilleur contrôle sur leur vie numérique.

Cette thèse explore l'utilisation distribuée de ces PDMS dans un contexte de réseau opportuniste, où les messages sont transférés d'un appareil à l'autre sans nécessiter d'infrastructure. L'objectif est de permettre la mise en œuvre de traitements complexes croisant les données de milliers d'individus, tout en garantissant la sécurité et la tolérance aux pannes des exécutions.

L'approche proposée utilise les environnements d'exécution de confiance pour définir un nouveau paradigme informatique, intitulé Edgelet computing, qui satisfait à la fois les propriétés de validité, de résilience et de confidentialité. Les contributions comprennent (1) des mécanismes de sécurité pour protéger les exécutions contre les attaques malveillantes visant à piller les données personnelles, (2) des stratégies de résilience pour tolérer les défaillances et les pertes de messages induites par l'environnement décentralisé, (3) des validations approfondies et des démonstrations pratiques des méthodes proposées.

Title: Privacy-preserving distributed queries compatible with opportunistic networks

Keywords: Decentralized data management, Crowd computing, Privacy protection, Opportunistic networks

Abstract: In today's society, where IoT and digital platforms are transforming our daily lives, personal data is generated in profusion and its usage is often beyond our control. Recent legislations like the GDPR in Europe propose concrete solutions to regulate these new practices and protect our privacy. Meanwhile, on the technical side, new architectures are emerging to respond to this urgent need to reclaim our own personal data. This is the case of Personal Data Management Systems (PDMS) which offer a decentralized way to store and manage personal data, empowering individuals with greater control over their digital lives.

This thesis explores the distributed use of these PDMS in an Opportunistic Network context, where messages are transferred from one device to another without the need for any infrastructure. The objective is to enable the implementation of complex processing crossing data from thousands of individuals, while guaranteeing the security and fault tolerance of the executions.

The proposed approach leverages the Trusted Execution Environments to define a new computing paradigm, entitled Edgelet computing, that satisfies both validity, resiliency and privacy properties. Contributions include: (1) security mechanisms to protect executions from malicious attacks seeking to plunder personal data, (2) resiliency strategies to tolerate failures and message losses induced by the fully decentralized environment, (3) extensive validations and practical demonstrations of the proposed methods.

Research for this thesis has been conducted within the PETRUS team, which is a collaboration between the DAVID laboratory from the University of Versailles-Saint-Quentin-en-Yvelines and the Inria Saclay Centre.

REMERCIEMENTS

J’adresse ma sincère gratitude envers toutes celles et ceux qui m’ont apporté leur aide et leur soutien tout au long de ces années de doctorat. Votre présence a eu un impact déterminant sur ma réussite, et je suis extrêmement reconnaissant d'avoir eu le privilège de vous compter à mes côtés pour cette étape significative de ma vie professionnelle.

En tête de mes remerciements, je tiens à exprimer ma profonde reconnaissance envers mes trois encadrants, Luc, Nicolas et Philippe. Votre expertise et votre exceptionnelle bienveillance ont insufflé une belle dynamique à tous nos échanges, et je suis très heureux de pouvoir poursuivre l’aventure avec vous encore quelques années. Un grand merci pour votre disponibilité et votre écoute lors des périodes difficiles, en particulier pendant les confinements de la crise Covid, où le manque de contact social et d’activité sportive pesaient grandement sur mon moral.

Ma reconnaissance s'étend ensuite aux membres de l’équipe PETRUS en commençant par les anciens doctorants, Dimitris, Julien Loudet, et Riad pour leur accueil des plus chaleureux et leur transmission du « spirit of the team ». Je remercie ensuite Julien Mirval et Robin, mes compagnons de route pendant ce doctorat, avec qui j'ai tissé des liens d'amitié précieux et dont les moments de convivialité que nous avons partagés au bureau ou ailleurs resteront parmi les plus beaux souvenirs de ma thèse. J’adresse également ma sincère gratitude à tous les autres membres de l’équipe : Ali, Floris, Guillaume, Iulian, Katia, Laurent, et Mariem qui m’ont accompagné et soutenu pendant toutes ces années avec une attitude toujours bienveillante. Enfin, je tiens à remercier chaleureusement Léo, un stagiaire exceptionnel qui m’a beaucoup aidé pour le développement de la plateforme de démonstration et avec qui j’ai eu la chance de faire un petit tour à travers le monde pour présenter nos travaux.

Je remercie également les membres du laboratoire DAVID, en particulier les doctorantes et doctorants : Baudouin, Hafsa, Jingwei, Lili, Perla, Rahma, Saeed, Saloua, Souheir, et Zoé que j’ai eu le plaisir de rencontré (ou de mieux connaître) et avec qui j’ai partagé d’agréables moments que ce soit sur le campus de Versailles ou lors de conférences.

Finalement, je tiens à exprimer ma profonde reconnaissance envers mes amis et ma famille, dont le soutien inébranlable et les encouragements constants ont été des piliers essentiels tout au long de mon parcours. À mes amis, je

souhaite exprimer ma gratitude pour leur présence dans les moments où j'avais besoin de m'évader et de trouver de nouvelles inspirations. À mes parents, frère, sœur, grands-parents, oncles, tantes, cousins et cousines, je suis infiniment reconnaissant pour leur engagement à mes côtés, leur soutien a constitué une base fondamentale de mon accomplissement. Enfin, je tiens à adresser un remerciement spécial à ma compagne, Eva, qui partage ma vie et a été d'une aide indéfectible, m'insufflant confiance et détermination à chaque étape de cette aventure.

Enfin, à toutes celles et ceux que je n'ai pas mentionnés mais qui ont contribué au bon déroulement de cette thèse, je vous adresse mes remerciements les plus sincères.

RESUME

La croissance exponentielle d'Internet et son influence sur la société ont conduit à une création de données en quantités sans précédent. En 2018, l'ICD [1] a estimé la sphère de données mondiales annuelle à 33 ZB (10^{21} octets), avec des prévisions de croissance à 175 ZB d'ici 2025. La périphérie d'Internet s'étend à un rythme beaucoup plus rapide que son cœur, et cette croissance ne devrait pas s'arrêter là, avec une estimation à près de 29,3 milliards d'appareils connectés pour 2023 [2]. L'omniprésence de l'Internet des objets dans notre vie quotidienne contribue à accélérer notre société vers l'ère du numérique, où les données forment le "nouveau pétrole" [3].

Bien que cette création de données ait apporté de grands avantages aux entreprises et aux gouvernements, elle a également soulevé de graves préoccupations en matière de sécurité et de confidentialité des données personnelles. En effet, de par nos interactions avec nos appareils informatiques (smartphones, tablettes, ordinateurs) et notre utilisation de services numériques (boutiques en ligne, réseaux sociaux), une quantité massive de données personnelles est collectée à notre insu. Ces données concernant des milliards d'individus sont souvent centralisées sur d'énormes serveurs (par exemple, Google, Amazon, Facebook), facilitant leur accès, leur traitement et leur partage. Cette centralisation des données dans un seul système d'information en fait alors un véritable "pot de miel" pour les pirates informatiques. Les cyberattaques qu'ils mènent visent à capturer le plus d'informations personnelles possible (noms, adresses, e-mails, mots de passe, dates de naissance, etc.) afin de les revendre à d'autres criminels, par exemple pour des campagnes de phishing. Ces violations de données sont devenues très courantes et touchent de nombreux secteurs d'activité [4].

Malheureusement, les cyberattaques ne sont pas le seul inconvénient de la centralisation massive des données personnelles. En 2013, Edward Snowden a révélé que le gouvernement américain, par le biais de la National Security Agency, organisait une surveillance massive de sa population avec la complicité des détenteurs de données [5]. De plus, le profilage basé sur les données personnelles (niveau de revenu, orientation politique, habitudes d'achat) pousse le vice encore plus loin, avec la possibilité de manipuler massivement la population. Par exemple, les publicités affichées sur les réseaux sociaux comme Facebook bénéficient d'un mécanisme de micro-ciblage pour adapter le contenu de la publicité au public ciblé. Les élections américaines de 2016 et brésiliennes de 2018 ont été fortement influencées par ces mécanismes de publicités ciblées sur Facebook [7].

Pour répondre à toutes ces dérives d'usage et renforcer la sécurité des données personnelles, l'Union Européenne a adopté un cadre législatif, le RGPD [8], entré en vigueur en 2018. Cette réglementation s'applique à toutes les organisations, publiques ou privées, qui traitent des données personnelles appartenant à des résidents européens. Elle réglemente la collecte, le traitement et l'utilisation des données, ainsi que leur période de conservation. Au niveau individuel, cette loi accorde le droit à la portabilité des données, un mécanisme qui permet à quiconque de demander une copie de ses données personnelles détenues par une entreprise ou une administration. Ce mécanisme de portabilité, suivant d'autres initiatives telles que Blue Button aux États-Unis [9] ou MesInfos en France [10], constitue un élément essentiel pour la réappropriation de nos propres données. Cependant, la portabilité des données seule ne suffit pas à retrouver un contrôle sécurisé et respectueux de la vie privée si les solutions techniques pour héberger ces données sont à nouveau hypercentralisées.

Ainsi, dans ce contexte de sensibilisation croissante à la confidentialité, la nécessité de reprendre le contrôle de ses données personnelles a conduit au développement de nombreux systèmes informatiques plus respectueux de la vie privée. À titre d'exemple, nous pouvons mentionner l'application de messagerie instantanée Signal [12], la boîte mail StartMail [13] ou encore la solution de stockage Tresorit [14]. Cette tendance a également suscité l'intérêt de la communauté scientifique, avec notamment la proposition de solutions de gestion de données personnelles décentralisées (PDMS) [15]. Conçus comme des assistants matériels ou logiciels centrés sur l'utilisateur, les PDMS permettent aux individus de gérer leur vie numérique avec des fonctionnalités allant de la collecte automatique de données à des tâches de traitement et de partage plus complexes, le tout dans un environnement sécurisé et facile à utiliser. Des solutions industrielles sont déjà disponibles, comme Cozy Cloud [16], qui implémente un PDMS pouvant être hébergé soit sur le cloud, soit directement sur un appareil personnel.

Afin de ne pas régresser par rapport aux systèmes centralisés, l'architecture décentralisée du PDMS doit permettre l'exécution de calculs croisant les données de plusieurs individus. À l'instar des travaux récents proposant des solutions techniques pour exécuter des requêtes distribuées sur les PDMS [17]–[19], cette thèse étudie les solutions permettant de distribuer des traitements de façon respectueuse de la vie privée et tolérante aux pannes. Cependant, contrairement à ces précédents travaux, nous nous intéressons spécifiquement au cas où les PDMS sont déployés sur des appareils personnels (par exemple, smartphones, PC ou objets intelligents) avec des communications non conventionnelles, c'est-à-dire avec des messages

envoyés d'un appareil à un autre via des canaux à courte portée (par exemple, Wi-Fi ou Bluetooth), formant ainsi un réseau opportuniste (OppNet) [20]. Notre objectif est triple : nous voulons construire une solution qui (1) soit à la fois générique et évolutif, permettant des calculs complexes sur les données de milliers d'individus, (2) soit sécurisée et respecte la vie privée des personnes impliquées, et (3) soit tolérante aux pannes malgré un environnement entièrement décentralisé, propice aux défaillances et aux pertes de messages.

Exemple de motivation. Le contexte inhabituel de cette thèse est particulièrement inspiré par le cas d'utilisation du projet DomYcile [21]. Notre équipe travaille en partenariat avec le département des Yvelines et la société HIPPOCAD (filiale du groupe La Poste) [22] pour proposer une solution de gestion de données pour les personnes âgées recevant de l'aide à domicile. Actuellement, près de 8 000 patients sont équipés d'une boîte informatique sécurisée où leurs dossiers médicaux et sociaux sont stockés. Ces boîtes ne sont pas connectées à Internet pour des raisons de coût, de sécurité et d'acceptabilité, et ne sont accessibles qu'au domicile du patient par les professionnels de santé. La plateforme DomYcile, mise en place par le département des Yvelines, est ouverte par conception, de sorte que des tiers (par exemple, des associations de patients, des organismes statistiques, des professionnels de santé) puissent proposer de nouveaux services d'intérêt pour les patients (par exemple, interroger des cohortes éphémères de patients consentants et leur fournir des conseils de santé).

DomYcile n'est évidemment pas le seul scénario d'application de cette thèse. En effet, contribuer avec ses données à des fins utiles pour la population est connu en Europe sous le nom d'*altruisme des données*. Nous envisageons que ce type d'interrogation de cohortes éphémères puisse se généraliser à d'autres situations telles que les *sondages opportunistes*. Voici une description de ces deux concepts :

Altruisme des données : Introduite dans le « Data Governance Act » de l'UE [23], cette proposition encourage les individus à donner leur consentement pour traiter leurs données personnelles à des fins telles que la recherche scientifique ou l'amélioration des services publics. La protection de la vie privée est primordiale dans ce contexte, car c'est un élément clé pour que les individus participent avec leurs données sensibles.

Sondage opportuniste : Lors d'événements accueillant un large public (par exemple, conférences, concerts, musées, matches), les participants pourraient contribuer avec leurs données (par exemple, centres d'intérêt,

nationalité, âge) à un traitement global pour améliorer leur expérience utilisateur en temps réel (c'est-à-dire adapter les services aux caractéristiques du public). La proximité des individus et de leurs appareils personnels rend l'utilisation des infrastructures de communication traditionnelles inutile, voire inappropriée, et ouvre la voie à de l'informatique opportuniste [24].

Contributions. Pour atteindre les trois objectifs énumérés précédemment et correspondre aux réalités de terrain de nos cas d'utilisation, cette thèse apporte les contributions suivantes :

1. La définition du paradigme de calcul Edgelet, une nouvelle architecture pour mettre en œuvre des traitements complexes sur des appareils personnels dans un environnement hautement distribué, sujet aux pannes et dépourvu d'infrastructure.

2. La proposition de mécanismes de sécurité robustes pour contrer les tentatives d'attaques malveillantes et protéger les données des individus impliqués dans les requêtes distribuées.

3. La présentation et l'analyse de trois stratégies de résilience différentes produisant des résultats valides et tolérant aux pannes et aux pertes de messages induites par l'environnement entièrement décentralisé.

4. La mise en œuvre de validations approfondies et de démonstrations pratiques des méthodes proposées.

Cette thèse est organisée en huit chapitres, commençant par l'introduction, dans lequel nous détaillons le contexte général, les motivations et les contributions.

Le chapitre 2 présente l'état de l'art et les connaissances préalables nécessaires pour comprendre le sujet, qui croise plusieurs domaines de recherche. Nous commençons par lister les différentes solutions de PDMS avant de détailler le contexte des réseaux opportunistes. Ensuite, nous étudions les architectures de calcul décentralisées pour comprendre les problèmes liés à la vie privée et à la tolérance aux pannes. Enfin, nous passons en revue les techniques de préservation de la vie privée pour les adapter à nos scénarios d'application.

Au chapitre 3, nous définissons le paradigme de calcul Edgelet, en commençant par ses caractéristiques et en déclinant le modèle de responsabilité associé. Nous poursuivons avec la formalisation du modèle de

requête et l'analyse d'une conception préliminaire naïve. Nous concluons le chapitre en posant le problème scientifique et en définissant les propriétés à satisfaire pour atteindre nos objectifs.

Le chapitre 4 aborde les problèmes de sécurité et de confidentialité liés aux requêtes distribuées. Nous détaillons une série de mécanismes pour protéger l'intégrité des requêtes et la confidentialité des données, couvrant ainsi l'ensemble du cycle de vie de la requête, c'est-à-dire de la déclaration et diffusion à la production des résultats finaux.

Le chapitre 5 est consacré à l'étude des stratégies d'exécution et leurs impacts sur la confidentialité et la validité. Nous constatons que, selon le type de traitement considéré, toutes les stratégies ne se valent pas et que des compromis peuvent être nécessaires.

Le chapitre 6 fournit une analyse quantitative des méthodes proposées avec une validation expérimentale des algorithmes sélectionnés.

Le chapitre 7 présente deux démonstrations du paradigme de calcul Edgelet, montrant l'intérêt pratique de notre approche.

Enfin, le chapitre 8 conclut cette thèse en résumant les principales contributions et en indiquant certaines orientations pour les travaux futurs.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The exponential growth of the internet and its influence on modern society has led to the creation of data in unprecedented quantities. In 2018, the International Data Corporation [1] estimated the annual global datasphere at 33 Zettabytes (10^{21} bytes), with forecasts of growth to 175 ZB by 2025. The internet's edge expanding at a much higher rate than its core, this staggering growth is not likely to stop, with an estimate of nearly 29.3 billion connected devices by 2023 [2]. This pervasiveness of the Internet of Things (IoT) in our daily lives contributes to the acceleration of our society towards the digital age, for which data is the "new oil" [3].

While this data creation has brought great benefits to businesses and governments, it has also raised serious security and privacy concerns when it comes to personal data. Indeed, through the interactions with our devices (e.g., smartphones, tablets, computers, connected watches) and the use of digital services (e.g., web searches, online stores, social networks), a mass amount of personal data is collected without our awareness. These data concerning billions of individuals are often centralized on huge servers (e.g., Google, Amazon, Facebook) facilitating their access, processing, and sharing. This data centralization in a single information system makes it a real "honeypot" for hackers. The cyber-attacks they carry out aim at capturing as much personal information as possible (names, addresses, emails, passwords, birth dates, etc.) in order to resell it to other criminals, for example for phishing campaigns. These data breaches have become very common and affect many different business sectors [4].

Unfortunately, cyber-attacks are not the only downside of the massive centralization of personal data. In 2013, Edward Snowden revealed that the U.S. government, through the National Security Agency, was organizing massive surveillance of its population with the complicity of data holders [5]. Moreover, profiling based on personal data (e.g. income level, political orientation, shopping habits) pushes the vice even further, with the possibility to massively manipulate the population. For example, advertisements displayed on social networks like Facebook benefit from a micro-targeting mechanism to tailor the content of the advertisement to the targeted audience, a mechanism that has also been shown to be capable of disclosing much more information [6]. The 2016 United States and 2018 Brazilian elections were typically heavily influenced by targeted political advertisements on Facebook [7].

To respond to all these usage drifts and strengthen security around personal data, the European Union has adopted a legislative framework, the GDPR [8], which entered into force in 2018. This regulation applies to all organizations, public or private, that process personal data belonging to European residents. It regulates the collection, processing, and use of data, as well as their retention period. At the individual level, this law gives the right to data portability, a mechanism that allows anyone to request a copy of their personal data held by any company or administration. The latter, following other initiatives such as Blue Button in the United States [9] or MesInfos in France [10], is an essential building block for the re-appropriation of our own data. However, the data portability alone is not enough to regain a secure and privacy-preserving control if the technical solutions to host this data are once again hyper-centralized (e.g. Google Drive [11]).

Thus, in this context of growing privacy awareness, the urgent need to regain control of one's personal data has led to the development of many privacy-friendly systems. Examples include the instant messaging application Signal [12], the mailbox StartMail [13], and the storage solution Tresorit [14]. This trend has also aroused the interest of the scientific community, including the proposal of a decentralized architecture, the Personal Data Management System (PDMS) [15]. Designed as user-centric hardware or software assistants, PDMSs empower individuals to manage their digital lives with features ranging from automatic data collection to more complex processing and sharing tasks, all in a secure and easy-to-use environment. Industrial solutions are already available, such as Cozy Cloud [16], which implements a PDMS that can be hosted either on the cloud or directly on a personal device.

In order not to regress compared to centralized systems, the decentralized architecture of the PDMS must allow the execution of computations crossing the data from multiple individuals. Following recent works proposing technical solutions for executing distributed queries on PDMSs [17]–[19], this thesis focuses on how to distribute processing in a privacy-preserving and fault-tolerant manner. However, unlike these previous works, we are specifically interested in the case where PDMSs are deployed on personal devices (e.g., smartphones, PCs, or smart objects) with unconventional communications, i.e., where messages are sent from one device to another using short-range channels (e.g., Wi-Fi or Bluetooth), thus forming an Opportunistic Network (OppNet) [20]. Our objective is threefold: we want to build a framework that (1) is both generic and scalable, allowing complex computations on the data of thousands of individuals, (2) is secure and respects the privacy of the people involved, and (3) is fault-tolerant despite a fully decentralized environment, prone to failures and message loss.

Motivating example. The unusual context of this thesis is particularly inspired by the use case of the DomYcile project [21]. Our team is working in partnership with the French Yvelines district and the company HIPPOCAD (a subsidiary of the La Poste group) [22] to propose a data management solution for elderly people receiving home assistance. Currently, nearly 8,000 patients are each being equipped with a secure home box where their medical and social records are stored. These boxes are not connected to the Internet for subscription cost, security, and acceptability reasons and are only accessible at the patient's home by healthcare workers. The DomYcile platform, set up by the Yvelines district, is open by design, so that third parties (e.g., patient associations, statistical agencies, medical workers) can push new services of interest for the patients (e.g., querying ephemeral cohorts of consenting patients and delivering them healthcare advice).

DomYcile is obviously not the only application scenario of this thesis. In fact, contributing one's data to useful purposes for the population is known in Europe as *Data Altruism*. We envision that this type of ephemeral cohort querying may generalize to other situations such as *Opportunistic Polling*. Here is a description of these two concepts:

Data altruism: Introduced in the EU Data Governance Act [23], this proposal fosters data subjects to give consent to process their personal data for purposes such as scientific research or public services improvement (e.g., a health survey organized by Santé Publique France). Privacy protection is paramount in this context, as it is a key element for people to participate with their sensitive data.

Opportunistic polling: During events that welcome a large audience (e.g., conferences, concerts, museums, matches), the participants could contribute with their data (e.g., centers of interest, nationality, age) to a global processing to improve their user experience in real-time (i.e., adapting the services to the characteristics of the audience). The proximity of individuals and their personal devices makes the use of traditional communication infrastructures unnecessary, if not inappropriate, and paves the way for Opportunistic Computing [24].

Contributions. To achieve the three objectives listed earlier and to match the field realities of our use cases, this thesis makes the following contributions:

1. The definition of the Edgelet computing paradigm, a new framework for implementing complex processing on personal devices in a highly distributed, failure-prone, and infrastructure-less environment.

2. The proposal of robust security mechanisms to counter malicious attack attempts and protect the data of individuals involved in distributed queries.

3. The presentation and analysis of three different resiliency strategies that produce valid results while tolerating the failures and message losses induced by the fully decentralized environment.

4. The implementation of extensive validations and practical demonstrations of the proposed methods.

This thesis is organized into eight chapters, beginning with the introduction, the current chapter, in which we detail the general context, motivations, and contributions.

Chapter 2 presents the state-of-the-art and background knowledge needed to understand this topic, which crosses several research areas. We will start by reviewing the different PDMS solutions before examining the context of Opportunistic Networks. Next, we will study decentralized computing architectures to understand issues related to privacy and fault tolerance. Finally, we will review current privacy preservation techniques to adapt them to our application scenarios.

In Chapter 3, we will define the Edgelet computing paradigm, starting with its definition and declining the associated responsibility model. We will continue with the formalization of the targeted query model and the analysis of a first straw man design. We will conclude the chapter by defining the problem statement and the distributed system properties to be satisfied.

Chapter 4 addresses security and privacy issues for distributed queries. We detail a series of mechanisms to protect the integrity of queries and the confidentiality of targeted data, addressing the entire query lifecycle, i.e., from declaration and dissemination to the production of final results.

Chapter 5 is devoted to the study of execution strategies and their impact on privacy and validity. We will see that, depending on the type of processing considered, not all strategies are equal and that trade-offs may be necessary.

Chapter 6 provides a quantitative analysis of the proposed methods with experimental validation of the selected algorithms.

Chapter 7 presents two demonstrations of the Edgelet computing paradigm, showing the practical interest of our approach.

Finally, Chapter 8 concludes this thesis by summarizing the main contributions and giving some directions for future work.

This thesis is based on the three international publications presented respectively at CCGrid 2022 [25], PerCom 2023 [26] and EDBT 2023 [27].

2 Background Knowledge and Related Works

In this chapter, we will explore the intersection of four different areas of research: Personal Data Management Systems, Opportunistic Networks, Decentralized Computing Architectures, and Privacy Preservation Techniques. Our goal is to understand these technologies and contexts in order to subsequently integrate them into resilient, valid and privacy-preserving distributed computing solutions.

2.1 Personal Data Management Systems

Personal data is one of the most valuable commodities of the digital age. However, the excessive collection, massive centralization, opaque management and sharing of this data by companies has raised many concerns about privacy and security. As a result, there is a growing demand

for user-centric solutions that allow individuals to regain control of their personal data. Personal Data Management Systems (PDMS), Personal Clouds, Personal Data Stores, and Personal Information Management Systems are some of the names given to these solutions. They allow individuals to securely store and manage all types of digital content, whether it is official documents, personal photos, videos or IoT-generated data. By enabling individuals to manage their personal data responsibly, these solutions have the potential to revolutionize the way we interact with digital content and protect our privacy.

In this section, we will provide an overview of the different types of personal clouds, their key features, and the privacy they offer. The goal is to understand the limitations of each solution, starting from the most common ones like Google Drive [11] to the most privacy-friendly ones like PlugDB [28].

2.1.1 Standard Personal Clouds

Standard Personal Clouds refer to cloud storage solutions offered by various providers, such as Google Drive [11], Dropbox [29], and OneDrive [30]. These services allow users to upload their data on the provider's infrastructure and access it easily afterwards via personal devices such as computers and smartphones. In recent years, personal cloud storage has become a popular choice for individuals who wish to store and access their data from anywhere and from any device.

One of the main advantages of Standard Personal Clouds is their ease of use. These services typically offer simple and user-friendly interfaces for uploading and managing individual files. They also offer a wide range of features, such as file syncing, versioning, and collaboration. File syncing allows users to automatically synchronize their files across multiple devices, ensuring that the latest version of the file is always available. Versioning allows users to access previous versions of a file, making it easier to track changes and restore previous versions if necessary. Collaboration features allow multiple users to work on the same document simultaneously, improving productivity and coordination [11], [30].

These benefits aside, Standard Personal Clouds have important limitations, particularly with respect to privacy and data management. Indeed, when it comes to sensitive individual data, these online storage approaches are not the most suitable as the entire security model relies on the IT architecture implemented by the provider. Even if the data is protected by cryptographic mechanisms, the encryption keys remain in the possession of the provider,

which does not prevent him from decrypting and using the data for purposes other than those intended by the person to whom it belongs. These secondary usages constitute what is known as data monetization (through targeted advertising, for example) and are the business model of these "free" storage providers. On the technical side, Standard Personal Clouds suffer from data management limitations as file storage is far from being the only feature required for a full PDMS. The fact that users have to manually upload their data makes the task complex and time-consuming whereas an automatic data collection mechanism would allow them to efficiently extract their digital life from the source of their choice (bank, insurance, IoT sensors, etc.). Users should also be able to perform processing on their own data and thus benefit from the computing capacities of the servers on which they are hosted. Unfortunately, this essential feature is not possible in this personal cloud category.

2.1.2 No-Knowledge Personal Clouds

No-Knowledge Personal Clouds, also known as zero-knowledge or end-to-end encrypted personal clouds, offer an enhanced level of privacy and security. The key difference between No-Knowledge Personal Clouds and Standard Personal Clouds is that in the former, the user retains full control over their data, including encryption keys, and the cloud service provider has no access to the user's data. This means that even if the provider is hacked by a virus or malicious administrator, the user's data remains secure and private.

There are several No-Knowledge Personal Cloud solutions available on the market today. For example, Tresorit [14] provides a cloud storage service that uses end-to-end encryption to protect users' files, making it impossible for anyone, including Tresorit employees, to access the data without the user's permission. Similarly, SpiderOak [31] and Sync.com [32] offer comparable solutions, each ensuring that only the user can access their data. As mentioned before, the key advantage of these solutions is that they provide a high level of security for sensitive data. Users can store confidential information, such as financial records or health data, without fear of it falling into the wrong hands. In addition, No-Knowledge Personal Clouds can allow users to share and collaborate on files securely [32], without worrying about data leaks or unauthorized access.

However, No-Knowledge Personal Clouds have the same limitation as Standard Personal Clouds regarding data management. Indeed, there is still no automatic data collection mechanism and no possibility to run complex

processing from the server. Concerning the ability to perform processing, this "end-to-end encrypted" architecture is in itself a hindrance to the development of the feature (unless using homomorphic encryption, see Section 2.4.1). Consequently, whenever users want to perform cross-computations on their data, they have to repatriate it to their own personal devices, which makes them directly responsible for their security/privacy.

2.1.3 Privacy-Friendly Personal Clouds

Privacy-Friendly Personal Clouds are gaining popularity as individuals are becoming more aware of the privacy risks associated with centralized data storage. These solutions provide a secure and private space for personal data on an online cloud that is controlled and managed by a provider, with the major difference of offering multiple features for data management. For example, data collection modules enable users to automatically import documents and other data from online services directly into their personal space, facilitating the data collection process. Cross-data computations and advanced data sharing are other crucial features that allow users to analyze and share data with third parties in respect of their explicit consent. Examples of Privacy-Friendly Personal Cloud include Cozy Cloud [16], Digi.me [33] and Solid [34] which provide more or less the same services.

Unfortunately, the main weakness of Privacy-Friendly Personal Clouds is the strong security assumptions on which they are based. The provider's employees, especially the administrators, are assumed to be fully honest, and all parts of the code, from the underlying storage mechanism to all the applications and services running on top of it, must be trusted. Since the data is centralized on the provider's infrastructure, a successful attack due to negligence or corruption of one of the employees could result in the leakage of a large amount of data from several people.

2.1.4 Home PDMS

We define Home Personal Data Management Systems as all hardware and software designed to store and manage personal data directly in users' homes. These devices can range from simple external hard drives to more complex devices that combine storage, processing power, and networking capabilities. Unlike online cloud storage solutions, where data is stored on servers controlled by third-party companies, Home PDMSs keep data under the control of users. There are several different types of solutions available, some are software-based that can be installed on existing hardware, while others are specific devices designed for personal data management. As

software-based solutions, we can mention OpenPDS [35], DataBox [36], Personium [37], as well as self-hosted instances of Cozy Cloud [16] and NextCloud [38]. On the side of the dedicated devices, there are several solutions on the market with for example Amber X [39], Helixee [40] and Meet Lima [41] that can store hundreds of gigabytes of data while synchronizing with other personal devices. In short, these solutions provide the same features as previously described, such as data collection, cross-computing and data sharing while being under the physical control of the users.

The major advantage of these solutions lies in the complete decentralization of the architecture. Indeed, from an attacker's perspective, it is highly preferable to attack a centralized system where millions (or even billions for Google Drive) of individuals are involved rather than trying to plunder the data of a single individual stored on a Home PDMS. Nevertheless, users still have to trust the software or hardware providers for all the features included in their solutions, and this without having any formal guarantees about their security. And to go one step further, it must be considered that these devices may be subject to flaws due to their usage or their environment, such as the presence of malware, viruses or unsecured end-to-end connections. In summary, we can see that although these Home PDMSs bring great perspectives in terms of decentralized data management, there are still some challenges to overcome before being able to "blindly" store all our personal data.

2.1.5 Portable PDMS with Tamper-Resistant Hardware

We have seen that PDMS solutions relying on cloud-based architectures or dedicated devices have limitations in terms of privacy and security. To address these issues, some research projects, such as the Personal Data Server [42] and Trusted Cells [43], propose to extend the Home PDMS with tamper-resistant hardware to achieve a secure portable device. These solutions leverage the security properties of secure chips (smart-cards, secure micro-controllers or trusted platform modules) in order to host a minimal Trusted Computing Base (TCB) protecting the personal data. Concretely, they embed a DBMS engine, such as PlugDB [28], into a chip to provide tamper-resistant storage and computing resources. The data is stored encrypted on a hard disk or a memory card whose reading is regulated by the access control mechanisms of the DBMS. These solutions are applicable in many contexts such as health data [44]–[46] and text documents [47] (see also tutorials [48], [49]).

This approach has been proven effective for simple queries and secure cross-

computations. Indeed, SQL queries are supported thanks to the integration of a query evaluation and an access control engine on the PDMS running inside the secure chip [47], [50]. In the context of a network of Portable PDMS, a first way to achieve secure distributed computing is to leverage an untrusted cloud infrastructure to transfer encrypted data among nodes [51], [52]. However, the use of a central server has again some vulnerabilities, since it is possible for it to deviate from the original code and leak information. A second way is to use the trust placed in the portable PDMS hardware to build a safe and reliable aggregation chain [17], [53]. Based on a peer-to-peer network, this decentralized architecture can also be used to perform highly distributed queries on PDMS [19].

2.1.6 Conclusion

Among all the PDMS classes presented in this section, Portable PDMS with tamper-resistant hardware appears to be the most robust, combining both computational capacity, security, and privacy. Nevertheless, it should be noted that the latter solution is still limited in terms of extensibility and resiliency, making it not very adaptable to other use cases. Based on the refined PDMS architecture [15] and Trusted Execution Environment (see Section 2.4.4), very recent works [54], [55] have proposed to address the extensibility issue by demonstrating that a minimal TCB can be coupled with user-defined functions to extend usages while strongly limiting the risk of data leakage. Regarding resiliency, however, previous works do not make significant contributions to the fault tolerance of distributed executions on multiple PDMSs, either from hardware failures or message loss. One of the major objectives of this thesis is to bring an answer to this last question, by studying the extreme case of communications in Opportunistic Networks.

2.2 Opportunistic Networks

The concept of Opportunistic Networks (OppNets) is a natural evolution of several years of research on multihop ad-hoc communication systems [56]. Back in the 1990s, mobile ad hoc networks (MANETs) were introduced to provide connectivity among (mobile) devices when no pre-existing infrastructure is available. However, an underlying feature of MANETs protocols assumed that there is an end-to-end path between senders and receivers, which has proven to be inappropriate in real life when the mobility and availability of devices vary steadily over time. A couple of years later, the Delay-Tolerant Network (DTN) architecture [57] was proposed to address scenarios where devices are mostly disconnected and communicate only occasionally (scheduled or not). The first works on OppNets [20] appeared in

the same period and were based on similar principles, which explains why some researchers consider them as an instance of DTNs [58]. In this section, we first review the characteristics as well as the use cases of Opportunistic Networks, and then discuss the main challenges and limitations associated with these particular communication systems.

2.2.1 Characteristics Overview

When reliable infrastructures (such as cellular networks) are not available or appropriate, OppNets provide suitable communication mechanisms for mobile user devices lacking end-to-end paths between them [59]. To this end, they rely on short-range wireless communication capabilities (e.g. Wi-Fi or Bluetooth) as well as on the "store-carry-and-forward" principle [58]. As explained by Conti and Giordano [56], it is the mobility of people that creates the possibility of connecting parts of the network that were not originally connected. Thus, mobility is no longer seen as a constraint, but rather as an opportunity. When it comes to personal devices, OppNets are considered as people-centric approaches, where each individual contributes to the transmission of messages by physically carrying the buffered data to the next intermediary.

Over the years, researchers have proposed different routing strategies to improve the performance of OppNets in terms of latency, energy consumption, and storage. These strategies can be classified into three categories, namely replication, forwarding, and hybrid, as fully detailed in [60] and [59]. For purposes of illustration and understanding, here is a brief overview of three popular routing protocols:

Epidemic routing (replication) [61]. As the name implies, this protocol spreads messages across the network like an epidemic: as soon as two nodes are in transmission range, they exchange any messages that the other does not have. Each message is then replicated from node to node at each opportunistic device contact until it reaches its destination. As a result, this strategy maximizes the message delivery rate and minimizes message latency at the cost of significant network congestion and overhead.

First Contact (forwarding) [62]. Unlike the epidemic strategy that replicates messages endlessly, the First Contact protocol proposes to transfer messages to the first device that appears. More precisely, when a node carries a message, it waits to meet another node to forward the message and removes it from its own buffer. This inexpensive message propagation is therefore risky, resulting in high latency and poor delivery rates in some cases.

PRoPHET (hybrid) [63]. The Probabilistic Routing Protocol using History of Encounters and Transitivity attempts to find a balance between replication and forwarding: only the nodes most likely to transmit messages to their destination are selected with a limit on the number of replicas. To this end, nodes maintain a history of encounters to establish a vector of delivery predictability values that will be updated incrementally as devices make contact. In addition, the adjustable number of replicas allows for a trade-off between delivery success rate and resource consumption.

2.2.2 Main Applications

We have just seen that the first advantage of OppNets is to propose communication methods based exclusively on edge devices (i.e., without using any pre-established infrastructure), which opens the way to many application fields. The taxonomy presented below, derived from [64], presents six different and non-exhaustive contexts/applications.

1. Communication in Challenged Areas	2. Cellular Network Offloading
• Monitoring and tracking [65]: environmental sensors, terrestrial or marine animals. • Sparse network and inaccessible areas [66]: inhabited regions, underwater, mines or space. • Disaster areas and war zones: unavailability of the infrastructure.	• Large crowds saturating the operators: sports events, festivals, and demonstrations. • The physical proximity of individuals and their devices allows the local dissemination of information using in situ resources, thus alleviating cellular traffic [67].
3. Censorship Circumvention	**4. Proximity-Based Applications**
• Totalitarian governments or institutions can control the Internet and censor information. • By nature, OppNets provides a means of communication that bypasses all types of restrictions, relying solely on people's devices and their mobility [68].	• Group monitoring [69]: tourists staying connected to each other while visiting a museum or a city. • Mobile Social Networks [70]: co-location of nodes at a specific time and area allows for the deployment of tailored applications (e.g., recommendations, media sharing).
5. Opportunistic Mobile Sensing	**6. Opportunistic Mobile Computing**
• Most of our devices are equipped with various sensors which can also communicate with those in the environment when in transmission range [71].	• After data dissemination, a natural application of OppNets is the sharing of processing among edge devices, providing more complex and tailored services to users [24].

• The collection of these data enables the study of the interactions between humans and the environment.	• The use of local personal devices allows to optimize the resource consumption in a trustable and secure way.

2.2.3 Conclusion

Almost twenty years after the first works on OppNets [20], this concept is not yet very popular, which is mainly due to a lack of major applications. In fact, as evidenced by the survey of Trifunovic et al. [64], the "Quest for a Killer App" [72] remains a current challenge. Crushed by the explosive growth of connected infrastructure, ad hoc and opportunistic communication systems are struggling to find their way into the market, and to date, only a few applications have emerged (e.g., Uepaa! [73], FireChat [68]). Even the original use case of OppNets, to provide communication systems for challenged areas, is no longer relevant today with the emergence of new connectivity accesses such as SpaceX's StarLink [74]. Conversely, application scenarios based on proximity or data offloading are still of great interest and their lack of widespread adoption is explained here by technical constraints. Indeed, current short-range communication technologies, such as Wi-Fi and Bluetooth, are still not adapted to dynamic contact opportunity detection [64], [75]: the idle state of devices waiting for a connection is extremely energy consuming and recent technological improvements such as Wi-Fi Direct or Bluetooth Low Energy are inadequate because they require manual pairing. All these situations of inapplicability of OppNets constitute a vicious circle, as few deployments mean few feedbacks on the technology and therefore make investors reluctant. However, this has not prevented researchers from improving their protocols over the years by testing them directly on small ad-hoc test beds or among the few existing simulators [76] (e.g., ONE [77], ns-3 [78]).

Among all the use cases presented above, one of the most promising applications for OppNets is the Opportunistic Computing paradigm [24]. Pooling the resources of personal devices is a key element of the Internet of People vision [79], enabling the implementation of "people-centric sensing and computing" applications. This unusual context, where computing capabilities lie at the extreme edge of the network, directly in the hands of individuals, still raises many challenges. Indeed, how to design a distributed architecture capable of managing data flows and processing with fair use of energy resources of personal devices? How to avoid selfish behavior and malicious attacks? How to ensure system resilience in an asynchronous

communication context, where no assumption can be made on the message delivery time? In particular, this last question raises a fundamental issue for distributed systems: the impossibility of consensus with failures and asynchronous communications [80]. Recent works [81], [82] present results that seem to circumvent this impossibility, but again, performance validation beyond simulation and very small test beds is required.

2.3 Decentralized Computing Architectures

Having reviewed the different PDMS solutions and understood the context of Opportunistic Networks, we will now provide an overview of decentralized computing architectures existing in the state of the art. Our goal is to identify application domains and discuss issues related to privacy and fault tolerance. We will focus on five main paradigms namely Wireless Sensor Networks, Crowd Processing, Edge Computing, Peer-to-Peer Systems, and Federated Learning. Then, we will see that although these paradigms have different names and sometimes come from distinct communities, the concepts and their applications are often quite similar and face common challenges.

2.3.1 Wireless Sensor Networks

Wireless Sensor Networks (WSNs) are a type of network composed of small, low-power wireless devices equipped with sensors for data collection. WSNs were originally developed for environmental monitoring and tracking needs [83], but with the rise of the Internet of Things (IoT) [84], they have become equally relevant for many other applications, such as healthcare, home and industrial automation [85]. The devices communicate with each other wirelessly and the data collected by the sensors is usually transmitted to monitoring stations for analysis. The processing performed on these devices is then streaming queries [86], [87], the majority of which is performed at the end of the chain in a centralized manner.

Privacy concerns. For the original use cases that consider sensors for the environment or for animal tracking, there are few if any privacy issues. However, in an IoT context, these sensors are increasingly invading our daily life [2] and therefore the process of collecting and processing data can quickly become very intrusive. In this context, González-Manzano et al. [88] proposes PAgIoT, a privacy-preserving protocol that leverages the capabilities of IoT devices to perform aggregation algorithms in a decentralized manner. We observe that this decentralization of processing reduces the risk of exposure of personal data, which is an essential element of privacy preservation.

Fault tolerance concerns. Since the sensors are often deployed in harsh environments, they are vulnerable to damage, failure, and interference. Moreover, when the devices are mobile, as is the case with Vehicular Sensor Networks [89], the ephemeral nature of communications makes it difficult to organize distributed processing as many messages may be lost or delivered too late. In response, O'Keeffe et al. [86] proposes a redundancy system called Replicated Dataflow Graph and suggests a "routing constraint" mechanism to coordinate data sources and replicas. But as the authors explain, this does not totally prevent the occurrence of routing inconsistencies, although infrequent in practice. This work is not the only effort in this area (see surveys [90], [91]), and there are still challenges to be addressed to ensure both resilient and valid decentralized processing.

2.3.2 Crowd Processing

Crowd Processing refers to a family of computing paradigms that involve the active participation of large groups of people collaborating together to perform a task or solve a problem. The three main concepts, namely Crowd Computing, Crowdsourcing, and Crowd Sensing, are all based on the idea that individuals are increasingly connected and can participate in the enrichment of the digital sphere, from the sharing of captured data to the realization of individual micro-tasks. Crowd Computing is a concept defined in several ways in the literature [92]–[94], usually referring to the participation of a large number of users in a distributed computing system to achieve a specific goal, such as data analysis or machine learning. For instance, Murray and al. [92] define it as the combination of mobile devices and OppNets, which is precisely what we consider in this thesis. Crowdsourcing [95] is a more specialized form of Crowd Computing, in which tasks are outsourced to individuals, who are typically paid for their work. Crowd Sensing [96] involves the collection of data from sensors embedded in personal devices, such as smartphones or wearables. All these approaches are usually driven by centralized servers on which part of the processing is done.

Privacy concerns. Crowd Processing is inherently privacy threatening since personal data can be collected and shared with third parties. The issue is well understood by the community which has proposed numerous privacy preservation mechanisms [97]–[99]. Brahem et al. [100] propose a new approach that takes into account the users' own tolerance to the use of the data provided, so that the Crowd Processing system guarantees users the expected level of privacy. Indeed, a consent-based multi-task allocation strategy is essential to allow the reuse of crowdsourced data contributions between tasks while strictly respecting users' consent.

Fault tolerance concerns. Fault tolerance is also a challenge in Crowd Processing, particularly in systems that rely on significant user participation. If a non-negligible proportion of users drop out or fail to participate, then system performance may be compromised. To address this situation, resiliency techniques can be used, such as task replication [101] or incentives for user participation [102]. An important issue in this context is the number of participants contributing to the processing and the management of the collected data, which may be received in multiple replicas and therefore need to be deduplicated [103].

2.3.3 Edge Computing

The Edge paradigm (also known as Fog Computing [104]) has emerged with the goal of offloading services and computation close to data sources to make the cloud more responsive, scalable, and privacy-friendly [105]. One of the key benefits of this paradigm is therefore its ability to reduce latency, as data processing can occur in real-time, directly at the edge of the network. This is particularly important for many applications in the IoT context [106], such as autonomous vehicles [107] and telemedicine [108]. Another benefit of Edge Computing is the reduction in the amount of data that needs to be transmitted to a centralized cloud infrastructure, which can result in lower costs, network performance, and privacy preservation. This technology operates mainly in connected infrastructures with processing that can be performed in Cloudlets [104], [109]. These are small-scale data centers (also called Micro Data Centers) deployed at the edge of the network providing computational and storage resources as close to the devices as possible.

Privacy concerns. Although Edge Computing is, by design, more privacy-friendly than a purely centralized approach, many privacy challenges remain, such as data leakage or malicious manipulation of devices [110]. For instance, Zhao et al. [111] propose a composable service system that offers the ability to run machine learning algorithms directly on connected devices or in combination with cloud resources, then allowing to adjust the level of data exposure and reduce the risk of data leakage.

Fault tolerance concerns. Because devices are more prone to network outages and service interruptions than centralized cloud infrastructures, implementing a fault-tolerant system can quickly become an issue [112]. This is even more true in the context of Mobile Edge Computing [113], where more processing is performed on the devices rather than on the previously mentioned Cloudlets. To address this issue, Grover and al. [114] designed a reliable IoT-Edge architecture that replicates sensed data (if the IoT devices

fail) and a redirection mechanism (if the server is not available). Note that this replication may raise other privacy concerns, as multiple copies of the same data are exposed, increasing the risk of leakage in case of compromise.

2.3.4 Peer-to-Peer Systems

Peer-to-Peer (P2P) systems are based on the principle that all devices have equal capabilities, and they can both request and provide resources to the network. This results in a highly decentralized system, as each device can communicate directly with the others, without the need for central servers [115]. Since the early 2000s, P2P networks have been widely used for file sharing, content distribution, and communication applications (e.g., Napster, Gnutella, MSN Messenger). They have also been leveraged in distributed computing contexts to collectively perform tasks or solve problems (e.g., BOINC [116], the Berkeley volunteer platform for sharing computing resources, including the World Community Grid project [117]). These tasks are usually split into smaller tasks and distributed across devices. The outputs are then combined to form the final result. This approach can be used for a variety of applications, such as scientific computing, data analysis, and machine learning [18].

Privacy concerns. Complete decentralization of the computing architecture alone is not enough to preserve individual privacy. Indeed, the fact that nodes communicate directly with each other makes it possible to access and analyze each other's data, which can represent a significant risk of exposure of sensitive data. In response, [118] suggests a P2P model for location-based mobile applications using the well-known k-anonymity mechanism. More recently, Mirval et al. [18] propose a secure aggregation protocol based on a secret sharing scheme providing fundamental building blocks for the execution of statistical and machine learning algorithms.

Fault tolerance concerns. Resiliency is also a major concern in P2P systems, as nodes can join and leave the network at any time [119]. This can lead to instability since the loss of nodes affects network performance and thus the execution of processing. To address this, P2P systems use techniques such as replication [120] and self-healing [121] to ensure that the network remains operational. For example, in a data replication scheme, data is replicated across multiple nodes, so that if one node fails, the data can still be accessed from another node. Again, we see that replication is essential to ensure resiliency, but can be detrimental to privacy.

2.3.5 Federated Learning

Federated learning is a decentralized machine learning architecture that allows multiple servers/devices to collaboratively train a model without exchanging their raw data [122]. This approach has gained popularity in recent years due to the increasing need for privacy preservation and data decentralization. The concept of federated learning covers two broad categories of applications: on the one hand, "cross-silo" where a few organizations with large amounts of data want to collaborate to build a machine learning model while avoiding sharing their data (e.g., pharmaceuticals discovery [123]), and on the other hand, "cross-device" where a large number of user devices participate in the model refinement (e.g., mobile keyboard prediction [124]).

Privacy concerns. Federated learning is, by construction, more privacy-friendly than the standard approach of massive data accumulation in centralized architectures. However, as we have just seen with P2P systems, decentralization alone does not protect against all privacy attacks [122]. For instance, an attacker may try to infer training data based on the parameters of the received models [125]. To counter these attacks, Cyffers et al. [126] propose "Muffliato", an implementation of local differential privacy to reduce privacy leakage in a fully decentralized federated learning context (no central server). Unfortunately, the confidentiality advantages of this solution come at the cost of degraded data quality and therefore a loss of accuracy in the resulting models.

Fault tolerance concerns. In the cross-device context (closest to our work), it must be considered that devices may be disconnected or unavailable during the learning process. Indeed, as explained in [127], the diversity of hardware and network connections makes the federated network heterogeneous both in terms of computational and communication capabilities. It is therefore common that out of a large number of devices, only a small proportion are simultaneously active. To cope with this, Smith et al. [128] propose the MOCHA algorithm for federated multi-task learning and showed convergence of the method even in the presence of nodes dropping out at each iteration of the learning process.

2.3.6 Conclusion

This section covers a wide range of decentralized computing architectures by first examining their scope of application and then focusing on privacy and fault tolerance concerns. We have seen that, with the exception of P2P

systems, these decentralized architectures still partially rely on central servers, whether for processing or coordination. It is therefore mandatory to maintain these servers in order to guarantee reliable and secure services. We have also seen that although decentralized architectures are by construction more privacy-friendly than fully centralized approaches, they are not sufficient to protect against all confidentiality attacks. Finally, we found that implementing the fault tolerance mechanisms required for any field application is sometimes detrimental to privacy, especially in the case of data replication which increases the risk of data leakage. All these factors will have to be considered when designing our own decentralized computing architecture.

2.4 PRIVACY PRESERVATION TECHNIQUES

In this last section, we will review the various techniques available in the literature to perform (distributed) computations on personal data while preserving privacy. We will study four main approaches ranging from cryptographic mechanisms to hardware components: Homomorphic Encryption, Secure Multi-Party Computation, Local Differential Privacy and Trusted Execution Environments. Our objective is to understand the general functioning of these approaches as well as their respective advantages and disadvantages in order to later define our own privacy protection strategy.

2.4.1 Homomorphic Encryption

Homomorphic encryption is a cryptographic technique that allows computations to be performed directly on encrypted data. This means that sensitive data can remain encrypted while still being processed, thus preserving data confidentiality. The concept of homomorphic encryption was first introduced by Rivest, Adleman, and Dertouzos under the name of Privacy Homomorphisms [129]. There are three categories of homomorphic encryption schemes:

- **Partially homomorphic** encryption schemes only support one operation, such as multiplication or addition, on ciphertexts. For example, the RSA cryptosystem [130] is partially homomorphic for multiplication.

- **Somewhat homomorphic** encryption schemes support both addition and multiplication on ciphertexts, but for a limited number of times. For instance, some encryption schemes allow unlimited additions and one multiplication.

- **Fully homomorphic** encryption schemes, on the other hand, enable an unlimited number of operations both addition and multiplication. The first fully homomorphic proposal was made by Gentry [131] in 2009.

Unfortunately, the limitations on the operations of partially and somewhat homomorphic encryption schemes make them unsuitable for generic computations. For fully homomorphic encryption schemes, the main concern is performance and scalability. Although much work tends to mitigate this, it still takes several tens of seconds to encrypt one 16 bytes block [132].

2.4.2 Secure Multi-Party Computation

Secure Multi-Party Computations (SMC) are algorithmic processes that allow individuals to perform a joint computation on their data without revealing anything other than the final result. They were formally introduced in 1982 by Yao [133], who posed the question of how two millionaires could determine who is richer without disclosing their own wealth to each other.

We distinguish two main paradigms for SMC problems. The *ideal model* assumes that there is at least one trusted third-party among the participants, while the *real model* makes no such assumption [134]. In this thesis, where the application context is fully decentralized, we do not assume the existence of a trusted third-party and therefore discard all solutions based on the ideal model. Among the SMC methods of the real model, we can mention:

Garbled circuit [135]. The idea is to construct a boolean circuit composed of many logical gates that represents a desired computation, and then "garble" it in such a way that the parties can evaluate it without learning anything about each other's inputs. This garbled circuit is distributed to the parties, who use it to compute the function and obtain the result of the global computation.

Secret sharing [136]. This approach consists of dividing a secret into multiple shares (based on Shamir's secret sharing [137]) in such a way that no single party can view the original data without the cooperation of the other parties. The parties then use their respective shares to compute a function without revealing any information about their individual inputs.

There are also other approaches to SMCs, such as methods based on homomorphic cryptography [138], or gossip protocols [139]. Overall, SMCs provide a powerful framework for enabling secure computations in various real-world scenarios where data privacy is critical. However, as with

homomorphic encryption, they suffer from performance issues when scaling up with a large number of participants. For example, SMC adaptations for distributed databases, such as SMCQL [140], are usually limited in terms of the number of participants and operations supported (in SMCQL, only two parties are supported).

2.4.3 Local Differential Privacy

Local Differential Privacy (LDP) [141] is a widely-used technique in distributed computation that provides a high level of privacy protection for individuals participating in a global dataset with their sensitive data. The approach is an adaptation of the general differential privacy model proposed by Dwork in [142]. Unlike other anonymization techniques, such as k-Anonymity [143], l-Diversity [144], t-Closeness [145], which apply to the dataset itself, LDP applies to the data collection process. The privacy protection is introduced through the use of randomized algorithms that ensure that no individual's data can be inferred from the inputs of a differentially private dataset.

To explain the LDP intuition, let us take an example. Suppose we want to obtain statistics on the number of participants in a demonstration by asking the question: "Did you participate in the demonstration?". Participants are then asked to flip a coin without revealing the outcome. If the result is heads, the participant answers honestly, but if it is tails, he or she flips another coin and answers "yes" if it is heads and "no" if it is tails. This technique protects the privacy of the participants since it is impossible for an attacker to know whether their answer is honest or not.

Although the LDP provides robust and scalable mechanisms for privacy protection, it introduces by construction a trade-off between privacy protection and the utility of the data and therefore the accuracy of the results. Furthermore, it has been shown that the estimated error is linear with respect to the number of attributes [146], making LDP unsuitable for high-dimensional datasets.

2.4.4 Trusted Execution Environments

Trusted Execution Environments (TEEs) are tamper-resistant processing environment providing secure storage and computing capabilities [147]. They ensure three main security properties: isolation of the executed code, confidentiality of the manipulated data, and remote attestation, which is a mechanism to prove the identity of the code running within a TEE. Over the past decade, we have seen a wide deployment of TEEs in devices widely used

by the public, such as personal computers and smartphones, making them prime candidates for Personal Data Management Systems [15].

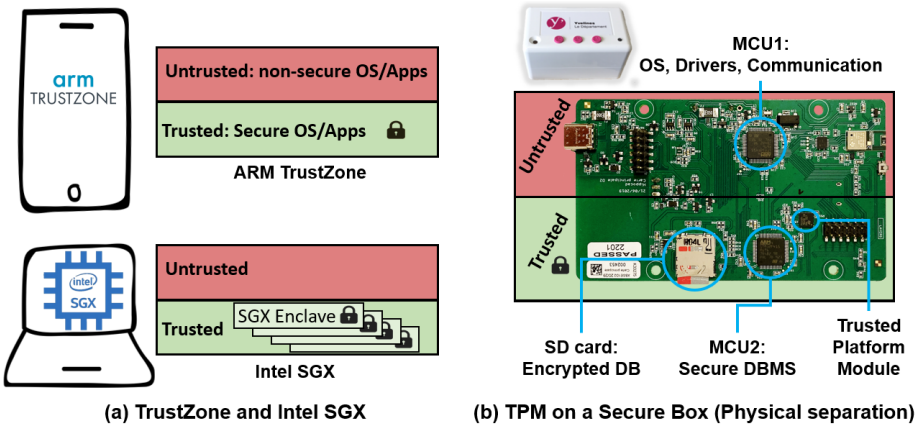

Figure 2.1: Examples of Trusted Execution Environments

ARM TrustZone. TrustZone [148] is a hardware-based security solution used in ARM processors that creates two separate "worlds": a secure and a normal one. These two parts are isolated from each other by partitioning the memory into secure and non-secure sections, and using special registers that can only be accessed when the processor is running in the secure state. Applications can then use the secure world to implement a TEE service, which performs a specific function, or a TEE kernel, which orchestrates several TEE instances by managing their memory, handling their communications, or providing them with APIs. Examples of TrustZone applications notably include TrustShadow [149] and Android Keystore [150]. More recently, Wan et al. [151] proposed a Rust-based TrustZone application SDK, called RusTEE, to help developers compile trusted applications with the enforced memory-safety features.

Intel SGX. Intel Software Guard Extensions (SGX) [152], [153] is a TEE technology introduced in 2013, which allows the creation of isolated software containers, called enclaves, that are protected from the operating system and the hypervisor. SGX provides data confidentiality and integrity protection by encrypting part of the memory and using specific parts of the processor. As already mentioned earlier, recent works [54], [55] have even demonstrated the practical application of a PDMS coupled to an SGX processor allowing to perform any kind of user-defined functions.

Trusted Platform Module. One of the precursors of TEEs is the Trusted Platform Module (TPM) [154], which is a hardware-based security module that provides a secure root of trust for computing systems. The TPM ensures

a high level of integrity by storing keys, passwords and certificates in a tamper-resistant environment on which cryptographic operations are performed. As explained in [147], the main limitation of TPM is that it does not provide an isolated execution environment for third-party, which reduces its functionality to a limited number of operations. Nevertheless, it is possible to combine the TPM with a CPU [155] or an MCU [53] (as illustrated in Figure 2.1.b) together with a minimal Trusted Computing Base to perform more complex processing.

Although TEEs are designed to be resistant to tampering and attacks, they are not completely immune. Indeed, these systems are regularly subject to new side-channel attacks [156]–[158], some of which are capable of breaking all security properties [158]. Thanks to their revelation by the scientific community, these attacks are addressed on a case-by-case basis via updates by TEE providers. Nevertheless, it is reasonable to consider the possibility of such attacks and to assume that a compromised TEE could behave in "sealed glass proof" mode [159], i.e., the confidentiality property is broken, but the isolation and attestation properties remain valid. Despite this, side-channel attacks are still complex to perform and usually require physical access to the TEE, making them less likely to occur on a large scale. It's important to note that honest TEEs cannot detect those that have been corrupted by side-channel attacks, as their behavior could still appear correct. This should therefore be taken into account when designing protocols based on TEEs.

2.4.5 Conclusion

As seen in this section, there are numerous techniques for combining data computation and privacy, each with its own advantages and disadvantages. In our case, the established objectives require a solution capable of (1) performing any kind of processing without compromising the quality of the results obtained and (2) scaling up with a large number of participants. According to our analysis, only TEEs seem to meet these constraints. The architecture must then include by construction the limitations of this technology, i.e., the possible side-channel attacks capable of disclosing all the personal data processed by the compromised TEE device.

3 EDGELET COMPUTING PARADIGM

The scenarios described in the introduction combine several characteristics. On the one hand, they require the execution of complex processing tasks on sensitive personal data. These tasks range from regular database aggregation queries to machine learning computations. On the other hand, they are executed in a highly distributed environment, prone to many failures, with communications between devices performed in an opportunistic manner.

To tackle the privacy protection, data management, and distributed system issues related to this environment, we define in this chapter the "Edgelet computing" paradigm. First, we describe the considered underlying architecture, its components, and related assumptions. Second, we propose a new model of responsibility adapted to this architecture, which specifies the role and obligations of each actor. Third, we discuss the data and query models targeted in this work and propose a straw man query execution plan which supports it. We then study the impact of the Edgelet computing context on this execution, which helps us to define three required properties to ensure the liveness, safety, and security of the query execution. The objective of this chapter is therefore to present the Edgelet Computing paradigm and to state the technical problem addressed.

3.1 EDGELET ARCHITECTURE

The idea of using the computational resources of personal computers or devices is obviously not new (cf. P2P architectures [115], e.g., BOINC [116]). More recently, [92] proposed to use resources from mobile personal devices, coupled with opportunistic communications to distribute opportunistically some large computations on a set of mobile devices. In this thesis, we push this idea a step further by seeking to share both the personal mobile device resources and the personal data of the device owner, as outlined in the scenarios presented in Chapter 1. To the best of our knowledge, this has never been proposed before probably because of the risks related to the protection of these personal data.

A game changer is the generalization of Trusted Execution Environments (TEEs) [147] at the extreme edge of the network: Intel SGX [153] is becoming ubiquitous on PC and tablets, ARM's TrustZone [148] on smartphones (Figure 2.1.a) and even Trusted Platform Module on smart objects (Figure 2.1.b). TEEs protect code and data from untrusted execution environments and from the devices' owners. More precisely, a TEE enforces (1) *data confidentiality*: data manipulated within a TEE node cannot be observed from the outside; and (2) *code integrity*: an attacker cannot influence the behavior of a program executing within a TEE. In this thesis, we assume that each personal device (edgelet) is equipped with a TEE guaranteeing to its owner the two previous properties as long as the device is not physically compromised. Indeed, as seen in Section 2.4.4, side-channel attacks compromising data confidentiality cannot be totally ignored, placing the device in "Sealed-Glass Proof" mode.

Regarding inter-device communications, the proposed architecture is built on Opportunistic Networks (OppNets). Indeed, the communication infrastructure considered in our scenarios is sidestepping any classical WAN infrastructure (e.g., Internet) for cost or energy constraints, lack of connectivity, security, or even freedom of expression concerns. The communications between edgelet nodes are short-range (e.g., Wi-Fi, Bluetooth) and asynchronous, i.e., there is no bound on the message transmission delay from a given edgelet node e_i to another edgelet node e_j. Connections among devices then form a non-connected time-varying graph as in traditional OppNets [81]. Note that various routing protocols from simple to more elaborated (and optimized) ones could be considered. However, the main focus of this study is not efficiency (e.g., resource consumption or query execution time), but rather feasibility. Hence, we consider for simplicity an epidemic diffusion of the messages, assuming that messages are transferred from device to device following a store-carry-forward communication protocol [61]. Other more optimized protocols, e.g., exploiting moving patterns of users [63], are considered as future work.

As mentioned earlier, we want to explore, in this thesis, the possibility to rely exclusively on secure personal devices to perform the required computations, i.e., without relying on any external infrastructure like central servers. Similar to P2P and cross-device federated learning architectures (see Sections 2.3.4 and 2.3.5), we want processing to be distributed across edge devices, without requiring the deployment or maintenance of centralized servers. Therefore, the execution of computations in the Edgelet architecture is achieved in a fully decentralized manner.

To summarize, the convergence between TEEs and OppNets, which we call

Edgelet computing, leverages secure personal devices to enable complex processing. Its architecture is characterized by the following three elements:

- ***Secure edgelet****:* Each personal device, called edgelet, is equipped with a TEE, which provides secure storage and processing capabilities.

- ***Opportunistic communications***: Edgelets communicate with each other through Opportunistic Networks whose messages are transmitted at short range with no time limit for their delivery.

- ***Fully decentralized execution****:* The processing of personal data is fully decentralized on edgelets and does not rely on central servers.

This new combination of secure hardware devices and the organization of a fully decentralized computations raises new issues in terms of work and responsibility distribution which are discussed in the next section.

3.2 RESPONSIBILITY MODEL

Responsibility models are usually introduced to help define the respective responsibilities of all actors involved in a given computing infrastructure. Such models guide judges, practitioners, and researchers when confronted with legal questions related to the protection of data and code. For instance, the *Shared Responsibility Model (SRM)* [160] states the responsibilities of cloud service providers and customers for securing all aspects of a cloud environment. To illustrate this, in an Infrastructure as a Service context, the customer is responsible for the data, application, and Operation System (OS) parts and the cloud provider for the rest of the infrastructure (virtualization, servers, storage, network), while in a Software as a Service (SaaS) context, the latter endorses also the responsibility of the application and OS.

The SRM is generic enough to apply to a large variety of application domains. Assuming personal data is managed in a SaaS context, the customer is the actor playing the data controller role[1] in the GDPR sense (e.g., a company, an administration, an NGO), and the cloud provider plays the role of data processor[2]. Yet, the individual herself is no longer in the loop after having given her consent to the data controller to process her data.

Conversely, in crowd computing applications managing personal data, the

[1] According to Article 4 of the EU GDPR, a data controller is the entity (person, organization, etc.) that determines why and how personal data is processed.
[2] A data processor is the entity performing the processing on the controller's behalf.

individual is at the heart of the infrastructure. However, we are not aware of any similar shared responsibility model adapted to crowd computing. Unsurprisingly, no one would agree to endorse the data processor or data controller responsibility in a fully decentralized computing context where each device is under the control of a distinct individual. The consequence is that crowd computing can handle use cases where the shared data is not really sensitive (e.g., sensor data like temperature or noise captured by a smartphone) but cannot tackle use cases involving sensitive personal data (e.g., medical data) that require tangible security guarantees. In this section, we define such a shared responsibility model, adapted to the Edgelet computing context, that we call *Crowd Liability Model (CLM)*.

Crowd liability conveys the idea that the data controller is the crowd (i.e., the result of the processing is expected to benefit, directly or indirectly, to each crowd member who agree together to the why and the how of this processing) and that there are as many potential data processors as they are crowd members. The corollary of this idea is that each crowd member is expected to do her best to honestly play the fragment of the data controller and data processor roles assigned to her, but the participation in the processing of some dishonest crowd members or of corrupted devices owned by honest crowd members cannot be precluded.

To translate this idea into a responsibility model, we split the data controller into two roles, (1) the *Recipient* which is the node selected to issue the processing and fairly disseminate[3] the result to the crowd, and (2) the *Regulator* (an external trusted entity or a set of crowd members) which assesses the honesty of the processing purpose and approves it on behalf of the crowd. Considering that crowd members usually do not have the technical skills to endorse the data processor role, we limit their responsibility to the delivery of accurate data as input for the processing and to the usage of a genuine TEE-enabled device to contribute to the processing (i.e., the crowd member is not liable for potential corruption of her own device but she becomes liable if she tampers with her TEE). Then, we introduce a *Trusted Assistant* role, which is played on behalf of the Edgelet node owners, and encompasses all technical principles embedded in each edgelet device to help the crowd members endorse the fragment of the data processor roles assigned to her. Basically, the Trusted Assistant is expected to guarantee the so-called *Computation honesty*, namely: (1) each decentralized execution

[3] The dissemination of the result being scenario-dependent, our study stops at the delivery of the final result to the Recipient.

strictly conforms to the QEP approved by the Regulator and (2) each honest crowd member confidently participates to the processing despite potential corruption of her own device and (3) each dishonest crowd member is defeated in her attempt to perform a massive attack.

Table 3.1 below transcribes the CLM with the distribution of responsibilities according to tasks and roles.

	Crowd Member	Recipient	Regulator	Trusted Assistant
Data integrity	X			
Edgelet integrity	X			
Result dissemination		X		
Purpose honesty			X	
Computation honesty				X

Table 3.1: Crowd Liability Model (CLM)

3.3 EDGELET QUERY MODEL

We consider distributed computations involving personal data hosted in (potentially large) sets of edgelet nodes from smartphones and tablets to more specific smart objects (see Chapter 7), like PDS [42], PDMS [54] or Databoxes [36]. Moreover, contrary to participatory sensing or sensor networks which focus on stream queries over elementary data, we consider rich data (e.g., healthcare folders, spending habits) and advanced processing (e.g., database statistics, data mining, machine learning). We assume that edgelets data can be queried as a shared database with a uniform schema. More precisely, each device may host a set of database schemas, typically one per application domain. The database schemas may be defined by a government agency (e.g., Ministry of Health), a private consortium (e.g., a group of banks and insurances) or an NGO. Consequently, for a given query expressed on a given database schema, the universe of edgelets data E can be seen as a horizontal partitioning of the corresponding global database.

The computations under consideration (called Query hereafter) must cope with the uncertainty inherent to the Edgelet setting, making the traditional database closed-world assumption [161] irrelevant. Thus, considering an open-world query model for edgelets leads us to introduce the notion of *snapshot-compliant query*.

Snapshot-compliant query: Given E the universe of edgelets data and Q a query targeting a dataset $D \subseteq E$, the representativeness of the snapshot D for Q is defined by a set P of predicates over elements of E (e.g., *age* > 65) and by a cardinality (e.g., $|D| = 2000$). We denote by $\zeta_Q(E)$ the set of all representative snapshots of E enforcing P and $|D|$. The Query Q is *snapshot compliant* if the result of Q considering any snapshot of $\zeta_Q(E)$ is equivalent for the Recipient.

In other words, a snapshot-compliant query, evaluated on any representative snapshot (enforcing P and $|D|$), will give a satisfactory result for the Recipient.

We consider that a query is expressed by a *Query Execution Plan* (QEP), which is a directed graph where vertices materialize the operators to be computed and edges represent the dataflow among them, with messages sent through the OppNet. The simplest form of a QEP is a tree with *Data Contributors* (DCs) at the leaves, that is the edgelets of crowd members who gave their consent to contribute to the query with their data[4]. Other operators of the QEP are *Data Processors*[5] (DPs), i.e., edgelets that contribute to the processing of the contributed data to produce the final result for the Recipient (R), the root of the QEP. Thus, the DPs either consume the outputs of a set of DCs or the outputs of other DPs.

Let us introduce the foundation for the query model by considering a straw man execution. To satisfy a snapshot-compliant query, we only need to define two types of data processors:

- The *Snapshot Builder* (SB) whose role is to build a representative dataset from the data transmitted by the DCs.

- The *Computer* (C) whose role is to perform the computation required by the query on the representative snapshot built by SB.

Then, the straw man QEP proceeds in three steps as illustrated in Figure 3.1.

1. Each Data Contributor DC_i sends its data to the Snapshot Builder.

2. SB builds a snapshot $D = \bigcup_1^n DC_i$ compliant with the set of predicates P and the cardinality $|D|$ and send it to the Computer.

[4] The consent management is an important question in the Edgelet context, addressed in Chapter 4.

[5] Data Processor refers here to an operator in a QEP, not to the Data Processor role as defined in the GDPR terminology.

3. C performs the computation on *D* and sends the final result to the Recipient (R).

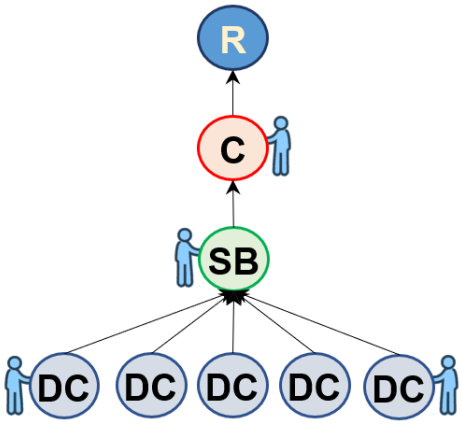

Figure 3.1: Straw Man Query Execution Plan

Note that, since we consider OppNets, we do not consider pipeline communication between edgelets because (1) it will generate too many messages, overloading the OppNet and (2) it will be very difficult to manage since there is no bound on transmission delays. Thus, we consider that a DP works in a blocking mode, i.e., it waits for all its input before processing it and producing the output. Similarly, a DC produces all its output in a single message. Consequently, any message is sent atomically through the OppNet, i.e., the payload is either totally received by the recipient or not at all.

3.4 Straw Man Execution Analysis

Obviously, this straw man execution does not answer any of the challenges posed by the considered context, namely data confidentiality, resistance to failures, and, as a consequence, execution validity.

Data confidentiality. As described in Section 3.1, the query execution is fully decentralized on edgelets that, even if secured by TEEs, can still be attacked (see Section 2.4.4). However, we do not wish to reduce the utility of the data by using local differential privacy. We cannot either use Secure Multiparty Computation techniques that would not scale due to opportunistic communications. An execution following strictly the straw man execution plan would therefore be disastrous in terms of exposed data since the potential effect of an attack on SB or on C would be to disclose all manipulated data. We will then have to circumscribe this risk by decomposing operators (e.g., using horizontal and/or vertical partitioning)

into sub-operators assigned to different edgelets. An operator should be decomposed if the generated sub-operators require fewer data to process, in order to minimize the impact of leakage in case of attacks.

Resistance to failures. Edgelets are personal devices and are therefore susceptible to failure and/or voluntary shutdown. In addition, these devices communicate via OppNets that do not guarantee bounds on communication delays. It will then be necessary to proactively introduce mechanisms to withstand these failures or communication problems. A resilient execution should therefore include backup or replication operators. The objective here will be to guarantee that the execution has a significant chance of success.

Execution Validity. The two previous points will lead to a much more complex QEP (with several SBs, several Cs as well as with backup edgelets and/or replicated operators) which could then result in an invalid computation if inconsistencies occur at some point. It will therefore be necessary to ensure that the execution is valid, either by adding synchronization mechanisms before the computation, or verification mechanisms after the computation.

The problem is quite complex because we will have to deal with these three dimensions at the same time, knowing that each one potentially influences the other two. For example, to resolve failures, we will need to add DPs, which will reduce security since data will be exposed on multiple edgelets. Similarly, to reduce the risk of data exposure, a DP needs to be split into multiple sub-DPs, which then increases the overall failure risk!

3.5 PROBLEM STATEMENT

To summarize the above analysis, we have to face three difficulties: First, the liability shift to the crowd must be carried out in a context where a few attacked devices can jeopardize the security of the whole system with no way to detect them. Second, reliable failure detectors cannot exist in OppNets due to the unpredictability of message delays, making it difficult to predict the time to build a snapshot from random contributors and to execute a query. The system liveness must then be guaranteed based on fault presumptions only and on the probability of success for queries associated to a deadline (i.e., a maximum time allowed for executions). Finally, the snapshot consistency must be preserved all along the query processing despite presumed faults and message loss between Data Processors in order to guarantee the system safety, i.e., a consistent result. We introduce below three properties that must be met together to tackle this problem.

Confidentiality (security property). Each edgelet must integrate mechanisms enforcing the CLM's Computation honesty on behalf of the edgelet owner, namely: (1) each decentralized execution strictly conforms to the QEP approved by the Regulator and (2) each honest crowd member confidently participates to the processing despite potential corruption of her own device and (3) each dishonest crowd member is defeated in her attempt to perform a massive attack.

Resiliency (liveness property). Given a probability of fault presumption p_f for any edgelet, a *query deadline*, and an expected probability of success p_s, a query Q must complete before the deadline with a probability greater than p_s, otherwise Q is aborted.

Validity (safety property). The Edgelet query execution result must be identical to a centralized query execution over at least one snapshot of ζ_Q(E). Formally, $\forall D_i \in \zeta_Q(E)$, $\exists D_j \in \zeta_Q(E) / Q_E(D_i) = Q_C(D_j)$, with Q_E (resp. Q_C) denoting the Edgelet (resp. centralized) execution of a query Q.

These properties are particularly challenging to tackle together given their mutual impact: Confidentiality is addressed in Chapter 4 while Resiliency is addressed in Chapter 5, studying its impact on Validity and Confidentiality.

4 CROWD LIABILITY ENFORCEMENT

The Edgelet computing paradigm introduced in Chapter 3 leverages the convergence between TEEs and OppNets to perform secure computations on personal data in a more flexible and scalable way than with local differential privacy, secure multi-party computation protocols, or homomorphic encryption. However, it comes with a rather specific Responsibility model called *Crowd Liability Model* (CLM). This chapter is devoted to the mechanisms and algorithms required to enforce this CLM.

The two cornerstones of the CLM are *Purpose honesty*, which deals with the approval by the crowd of the why and the how of the processing, and *Computation honesty,* which ensures the confidentiality of the processing during its decentralized execution. These two principles are respectively addressed in sub-sections 4.2 and 4.3. However, since the CLM defines new roles and responsibilities and relies on specific security assumptions, a dedicated threat model must be defined first.

4.1 DEDICATED THREAT MODEL

A dedicated threat model is required to capture (1) the shift of responsibility from a usual central entity (i.e., the data controller in the GDPR) to the crowd and (2) the TEE trustworthiness. We define this model as follows.

Ingenuous Recipient.

- *Role*: initiates the processing of a Query Execution Plan and receives the final results. Does not take part in the execution.
- *Played by*: depending on the use case, one or more crowd members, medical workers, statistical agencies, etc.
- *Trust:* We do not want to impose the Recipient to be equipped with a TEE-enabled device to avoid reducing the targeted use-cases, explaining why we exclude it from the actual processing of the QEP. However, we do not question the good faith of the Recipient. Thus, even if it receives the

results of the processing in clear text, we do not consider inference attacks from its side (i.e., crossing the results of multiple queries).

Wolf in sheepfold Participants.

- *Role*: contributes with data (*Data Contributor*) and/or processing power (*Data Processor*) in a Query Execution Plan.
- *Played by*: any participant equipped with a TEE-enabled device.
- *Trust*: as said in Chapter 3, side-channel attacks on a TEE-enabled device cannot be totally precluded despite their complexity (requires tampering with the device). A compromised TEE behaves in a "sealed glass proof" mode [159], where code integrity is preserved but data confidentiality is lost. We assume a large majority of honest participants (the lambs) and a few "sealed glass-proof" ones (the wolves).

Regulator.

- *Role*: reviews and approves the processing to be performed. This role is not devoted to the query execution itself.
- *Played by*: either an external entity (e.g., a privacy regulatory agency like the CNIL in France) or a representative group of crowd members engaged in a collective validation process.
- *Trust*: full.

4.2 PURPOSE HONESTY

In this section, we focus on the initialization of queries and how the Recipient can prove its honesty. To this end, we propose to build on the manifest-based framework [17], [53], so that the recipient can declare the why and how of processing. Here is its implementation in our context:

First, the Recipient specifies a manifest describing the query to be performed. This manifest consists of four elements: (1) the general purpose of the processing expressed in natural language, (2) the Query Execution Plan, (3) the set of representativeness predicates P of the targeted dataset D as well as its cardinality $|D|$ and (4) the source code of the operators to be executed on each edgelet node. Next, the Recipient submits the manifest to the Regulator which verifies its compliance with the expected privacy practices. Finally, the manifest is signed by the Regulator and returned to the Recipient. This certified manifest will be used for the edgelet assignment protocol presented in the next section, after which the query manifest will be ready for dissemination in the Opportunistic Network.

Let us now see how the threat model and the manifest-based framework

translate into the motivating example given in the introduction, the DomYcile project [21]. In this example, illustrated in Figure 4.1, we consider that a group of medical doctors (Recipient) wants to query a cohort of consenting elderly patients to obtain statistical results in the spirit of [162] (Share EU project [163]). Assuming that each patient is equipped with a secure box (see Figure 2.1.b), the query deployment proceeds as follows.

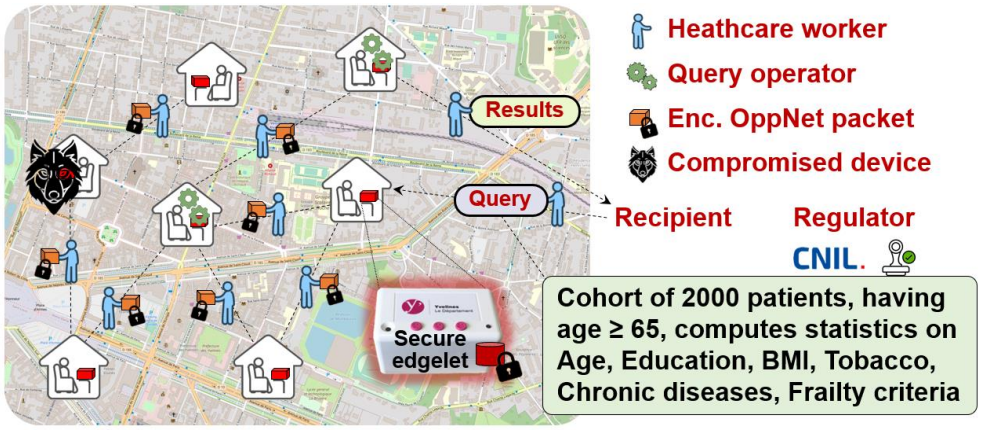

Figure 4.1: Edgelet query in the DomYcile project

First, the Recipient specifies the manifest containing the four elements mentioned above, including the general purpose: "Cohort of 2000 patients, having age ≥ 65, computes statistics on Age, Education, BMI, Tobacco, Chronic diseases, Frailty criteria". Second, the Recipient sends the manifest to the Regulator, e.g., the CNIL (French regulatory agency), which sends back a signed version if approved. Third, QEP operators are randomly assigned to some edgelets by the Recipient (see Section 4.3.2) making the query manifest ready for dissemination. The broadcast is performed by healthcare workers who implement an OppNet. The messages are transmitted from their smartphones to the boxes via Bluetooth using the store-carry-forward strategy. Patients willing to contribute their data to the query will then act as Data Contributors. Edgelets assigned by the Recipient will themselves act as Data Processors. The challenge is finally to protect the confidentiality of the computations despite potential attacks on the edgelets (wolfs), which is the topic of the next subsections.

4.3 Computation Honesty

4.3.1 Global and local integrity of the processing

Now that the certified manifest is disseminated to all edgelets in the network, we want to ensure the first two points of the CLM's Computation honesty: (1)

each decentralized execution must strictly conform to the QEP approved by the Regulator and (2) each honest crowd member must be able to confidently participate to the processing despite potential corruption of her own device. In other words, the question is how to verify that the execution is done in the same order as specified by the certified QEP and how to ensure that the processing performed on the data does not deviate from the original code?

To achieve this, we leverage the code integrity property provided by the TEE to build a Trusted Assistant, a logical entity taking responsibility for the Computation honesty on behalf of the edgelet owners. Concretely, the Trusted Assistant runs on each edgelet node a piece of code, called Core hereafter, which is part of the TEE Trusted Computing Base, that is a code base guaranteed genuine at boot time. Using remote attestations [17], [164] (cryptographic proof of the authenticity of the TEE), the Core enforces point (1) of CLM's Computation honesty by guaranteeing that the predecessors of Data Processors are legitimate and produce valid intermediate results. In addition, the Core attests the genuineness of the QEP operator's code assigned to the participating edgelets by verifying the signature of the Regulator on the certified manifest. Note that even if the edgelets' operating system is corrupted by malware or viruses, the isolation property of the TEE ensures that the executed code cannot be altered. Hence, point (2) of CLM's Computation honesty is also enforced.

These two integrity guarantees provided by the Trusted Assistant via the TEEs reinforce the crowd's confidence in the Edgelet computing architecture. As a result, each individual can freely consent to contribute her sensitive data with the reassurance that the decentralized execution will follow precisely the precepts indicated by the certified manifest. As we will see below, the next challenge is to protect executions from dishonest crowd members.

4.3.2 Resistance to massive attacks

Point (3) of CLM's Computation honesty states that each dishonest crowd member is defeated in her attempt to perform a massive attack. As mentioned before, we do not exclude the possibility of side-channel attacks on edgelets compromising the confidentiality of the manipulated data. Our objective is then to empower the Trusted Assistant with mechanisms that minimize the risks of massive data leakages in this particular case.

The Recipient is in charge of initiating the query, that is, assigning the QEP operators to the participating edgelets, and then launching the processing. However, the threat model does not make any assumptions about the

integrity of the Recipient(s). Even if we do not question his good faith, his information system may have been corrupted by some attacker. Thus, the task of assigning operators to edgelets must be protected to prevent a critical operator from being assigned to a corrupt accomplice edgelet, e.g., the operator manipulating the data of a targeted contributor or the operator manipulating a maximum amount of sensitive data. We therefore propose a random assignment protocol auditable by the Trusted Assistant.

Random assignment. We assume that the set of all crowd members' edgelet is known by the Recipient (e.g., they register to join a community) and that their ID form a hash ring (as in a Chord DHT). We also assume that each crowd member tacitly consents to contribute to a query execution, which is far less engaging than consenting to contribute their personal data to the query. Under these assumptions, the assignment protocol is the following.

1. The Recipient computes a hash of the manifest (signed by the Regulator and publicly known) as a seed for the random process and assigns the first operator to the edgelet having the ID immediately greater than this hash[1]. The first hash is rehashed to assign the second operator and so forth until all operators have been assigned.
2. The Trusted Assistant, which implements Core on all edgelets, verifies this chain of hash in order to detect any fraudulent assignment for the operator intended for them. Assuming that each edgelet knows at least the ID of its predecessor in the ring, it is sufficient to check that the hash is strictly larger.

How does this assignment work when successive hashes lead to collisions? Note that this problem is frequent when few edgelets nodes are present in the ring, and becomes highly unlikely otherwise. However, assigning multiple roles to a single node is undesirable for both resiliency and privacy reasons, hence the need for a countermeasure. When a collision occurs, we propose that the Recipient assigns the direct successor of this node in the ring, the latter having a lower probability (squared) of also being in collision. The Trusted Assistant can still detect a fraudulent assignment, but this time each edgelet must know the IDs of its two predecessors. The proposed method is therefore adjustable to any dimension of the network, it will just be necessary to increase the number of known predecessors.

[1] This assignment is blind; since it does not take into account the fact that edgelets may be down or unavailable. As we will see in Chapter 5, QEPs must provide resiliency mechanisms to anticipate these possible failures.

Next, let us try to restrict the leakage to dishonest crowd members, those for which the edgelet is physically attacked. Trivially, if no Data Processor is compromised, the TEE confidentiality property guarantees that no sensitive data can leak by construction. However, the distributed executions we are considering require intermediate results to be transmitted between the Data Processors (e.g., the representative snapshot *D* sent by the Snapshot Builder to the Computer). The messages sent in the OppNet then need to be encrypted to counter any malicious interception (e.g., by the untrusted smartphone of a healthcare worker).

Messages encryption. We assume that the QEP transmitted by the recipient contains the certificates of the assigned data processors, consisting of the IDs of the nodes in the DHT and their public encryption keys. Since the QEP is constructed statically, each node can easily encrypt its output based on its successors in the execution tree. Thus, when an edgelet e_i needs to send a message *m* to e_j, it generates a symmetric key k_s to encrypt its message and use e_j's public encryption key pk_j to encrypt k_s. The packet sent in the OppNet is then: {enc(m,k_s), enc(k_s, pk_j)}. Following this procedure, all messages in transit in the network are encrypted, making them unreadable to anyone other than their recipients. Therefore, since no cryptographic material is ever shared among edgelets and the TEE integrity property still holds even in sealed-glass proof mode, the potential leakage is reduced to the data processed by a compromised edgelet. Note also that any change in the operators ordering would make the messages indecipherable and the execution would fail, reinforcing point (1) of the CLM's Computation honesty.

Finally, we need to restrict the amount of data manipulated by each edgelet, so that in the event of a physical attack on assigned edgelets, leakage is limited to a small proportion of the data required for a query.

Horizontal and vertical partitioning. We observe that computations of interest are often distributive (e.g., MapReduce, Spark), enabling the decomposition of processing into sub-operators. Thus, we propose to distribute the operators of the Data Processors (Snapshot Builder and Computer) among different edgelet nodes. This decomposition can help minimizing the amount of data exposed at each edgelet by horizontally partitioning the dataset. This can also preclude the concomitant exposure, in the same edgelet, of data items that become sensitive when combined (e.g., a quasi-identifier) by vertically partitioning the dataset. Note that such distributive executions can also help minimizing the workload (e.g., when energy consumption matters) by exploiting the inherent Edgelet computing parallelism. Figure 4.2 presents both types of partitioning, horizontal and

vertical, on the query example of Figure 4.1. Each Data Contributor performs a hash function of its ID to select the Snapshot Builder to send its data to, so that each partition processes only a fragment of the dataset (here, a tenth) with different Computers depending on the statistics to be computed (each one only sees the attributes strictly necessary for its computation). Note that a Computing Combiner operator must be added in the QEP to combine the outputs of all sub-operators.

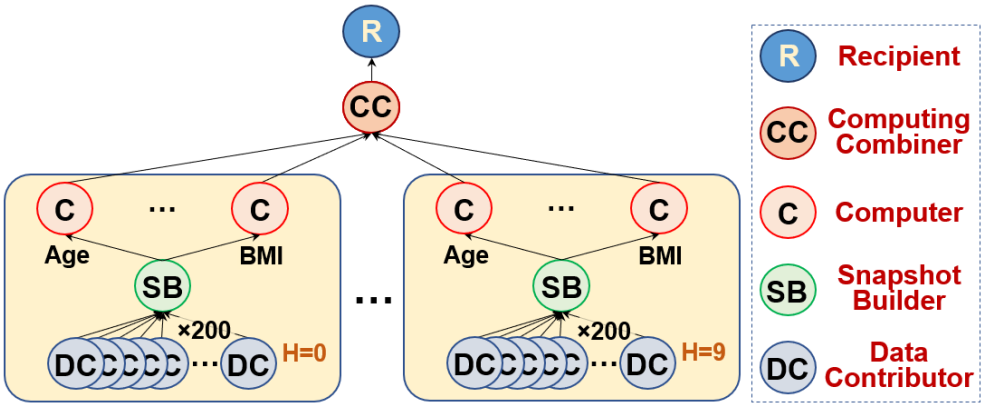

Figure 4.2: Horizontal and Vertical Partitioning

Although these partitioning strategies are decided by the Recipient in the design of the QEP, it is the responsibility of the Regulator to approve them. Depending on the application context, the privacy criteria selected may be different (e.g., medical records, spending habits) and involve a more or less distribution of the data. Note that since each individual is free to consent to contribute her data, it is in the Recipient's interest to design a privacy-friendly QEP in order to encourage people to participate.

4.4 CONCLUSION

The Edgelet computing paradigm involves the implementation of a unique and unusual responsibility model, the Crowd Liability Model (CLM). In this chapter, we have seen that the CLM must first be translated into a threat model in order to consider potential attacks on TEES and thus guarantee the Purpose and Computation honesty.

For Purpose honesty, we proposed a protocol based on the manifest-based framework [17] that establishes trust between the crowd members and the Recipient(s) through the Regulator's action. With this framework, any entity, public or private, can launch a query targeting the data of thousands of crowd members while certifying its intentions.

For Computation honesty, we leverage the properties of the TEEs to empower the Trusted Assistant (the logical entity) with several mechanisms to ensure the security of executions. We show that processing integrity is preserved throughout the decentralized execution and that despite confidentiality attacks on edgelets, leakage is limited to only the data handled by the corrupted devices, and in adjustable proportions thanks to horizontal and vertical partitioning.

5 EXECUTION STRATEGIES

The execution plans presented until this chapter are relevant from a logical point of view but are not resilient to failures. Indeed, the slightest failure or unavailability among the assigned edgelets results in the complete failure of the query. Our objective in this chapter is then to propose execution strategies that make Edgelet processing resistant to failures and to message losses in the Opportunistic Networks context.

In the following subsections, we present three alternative implementation strategies for enforcing the *Resiliency* property and discuss their impact on *Validity* and *Confidentiality*. First, we consider an extension of traditional resilience solutions based on backups (Section 5.1). Second, we present an alternative approach, in which, instead of replicating input data contributions, we tolerate an open (over-)set of data contributors typical of the edgelet context and take advantage of this "overcollection" of data to ensure Resiliency (Section 5.2). Third, we present a "hybrid" approach in which the backup and overcollection modes can coexist for better efficiency (Section 5.3). Finally, we conclude the section by comparing the different strategies in terms of their respective scope (Section 5.4).

5.1 BACKUP-BASED EXECUTION STRATEGY

In this section, we take a conservative approach to Resiliency, recovering from failures in a general way, independent of query plans and study its impact on the enforcement of the Validity and Confidentiality properties.

5.1.1 Enforcing Resiliency

No reliable failure detector exists in our context and every Data Processor (SB, C and CC in Figure 5.1) is a potential Single Point of Failure (SPF). In the Backup-based approach, we simply try to recover from failures, whatever the Data Processor presumed faulty, the benefit of which being to make the handling of Resiliency independent of the form of the QEP. We use timeouts to presume faults and secure the execution of all SPFs by means of backups, as usual [165].

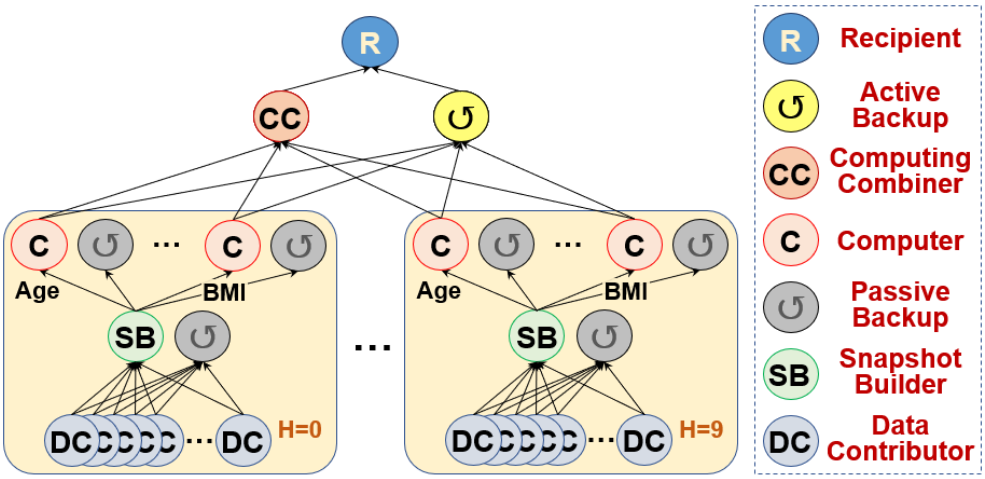

Figure 5.1: Resiliency based on the Backup strategy

We distinguish Passive and Active Backups. A Passive Backup replicates the input data of its corresponding SPF, called primary, and is activated and processes this data only in case the primary is presumed faulty. Thus, the data transferred to a Passive Backup is not exposed since it remains encrypted until the backup node is actually required. Conversely, an Active Backup executes in parallel with its primary. Despite a reduced latency, Active Backups incur a higher resource consumption and higher data exposure. Consequently, all SPFs are passively replicated (see Figure 5.1), except the Computing Combiner which must be actively replicated; otherwise, the Recipient would be forced to take part in the processing, at least to activate the Computing Combiner Backup(s), which would damage the Ingenuous Recipient assumption (see Section 4.1).

Figure 5.1 presents the Backup strategy for the QEP of the motivating example with horizontal and vertical partitioning. For this execution plan to be fault tolerant, each SPF in each partition must survive (i.e., Snapshot Builder and Computers), either by means of a primary node or one of its

backups (note that in the figure, only one backup per SPF is shown). The number *b* of backups per primary node in each partition is then determined by the inequality $\left(1 - {p_f}^{(1+b)}\right)^{|SPF| \times n} \geq p_s$, with p_f the probability of fault presumption, p_s the expected probability of success, and $|SPF|$ the number of SPFs in each of the *n* horizontal partitions of the QEP (n=10 in Figure 5.1). As we will also see in the other strategies, the Computing Combiner is considered separately from the other Data Processors due to its particular role as a proxy for the Recipient. Following the same principle, the number of backups b_{cc} for the Computing Combiner results in the inequality $\left(1 - {p_f}^{(1+b_{cc})}\right) \geq p_s$.

We assume that a start date T_s is set in the query manifest from which all Data Contributor edgelets start responding to the query. This date is then known by the Data Processor edgelets assigned for the query. Given the maximum delay δ for a sent message to arrive at its destination node (with a very high probability, close to 1), the maximum execution time of the query to successfully terminate can be estimated, which is referred to as the query deadline calibration. To this end, intermediate timeouts must be set at each Data Processor node so that the backups of their predecessor node(s) can be appropriately activated at runtime. For example, in Figure 5.1, the Computing Combiner and its active backups must allow sufficient time for all predecessor Computers to recursively activate the Snapshot Builder backups. The intermediate timeouts can be obtained using a simple recursive calculation based on the following principles, illustrated with an example (see Figure 5.2).

- For each data processor, a timeout Δ is associated with each of its predecessor nodes, whether primary or backup. When this timeout Δ expired, a fault of the corresponding predecessor node is presumed and an activation message is sent to the appropriate backup node (the first backup of a primary if the primary is presumed faulty, the second backup if the first backup is presumed faulty, etc.). After the activation message has been sent, only the first message received from either the presumed faulty primary or any of its activated backup, will be considered and processed.

- Primary and active backups nodes follow a push message pattern for all their successors (whether primary or backup nodes), meaning that messages are sent to successor nodes on their own initiative. Conversely, passive backups nodes follow a pull message pattern, i.e., messages are sent after a successor node's activation message is received.

Level 0: Snapshot Builder and its backups. Since the Data Contributors are never recalled, there is no timeout defined at the Snapshot Builders level and their backups.

Level 1: Computer and its backups. The Computer nodes (primary and backups) should receive their input data from the Snapshot Builder primary node after a maximum time delay of ~2δ. Indeed, the Snapshot Builders and their backups are supposed to receive the messages from the Data Contributors after a delay δ. Assuming that the data processing time of a node is negligible compared to the message latency, an additional maximum delay of δ is needed for the Snapshot Builder to transmit its result to the Computers. A first intermediate timeout at the Computer is hence fixed at $\Delta_1=2\delta$, at which the first backup of the Snapshot Builder will be activated. Recursively, as we expect the result of this first backup to be received after a maximum additional 2δ delay, we can define the second intermediate timeout (activating the second backup) at $\Delta_2=\Delta_1+2\delta$, and so on for the next backups.

We can generalize this to obtain the timeout Δ_b for any passive backup node in the tree. Indeed, this timeout is determined by adding 2δ (activation and response delays) to the Maximum Execution Time of a given Backup node (*METB*). At tree level l+1, *METB* is equal to the sum of:

- The consecutive[1] activation time of all direct predecessor backup nodes and their corresponding response time: $b \times 2 \times \delta$, with b the number of backups nodes.

- The recursive runtime of these backup nodes if they are not yet provisioned. This value depends directly on the number of levels l in the sub-QEP (i.e., the tree height): $b \times METB(l)$

Thus, $METB(l+1) = b \times \big(2 \times \delta + METB(l)\big)$

Similarly, the Maximum Execution Time of a Primary node (*METP*)[2] at tree level l+1 is equal to the sum of:

- The emission time of its direct predecessor primary node and its recursive runtime: $\delta + METP(l)$

[1] To protect privacy (i.e., by minimizing the number of data exposed), we choose to activate the backup nodes only when necessary, one after the other.
[2] *METP* also applies to active backups since they run in parallel with the primary node.

- The successive activation time of the direct predecessor backup nodes and their corresponding response time: $b \times 2 \times \delta$

- The recursive runtime of these backup nodes if they are not yet provisioned: $b \times METB(l)$

Thus, $METP(l+1) = \delta + METP(l) + b \times (2 \times \delta + METB(l))$

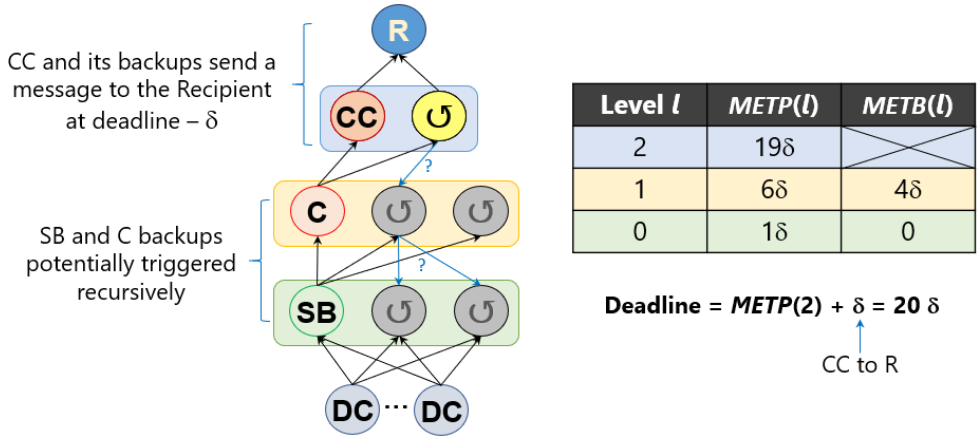

Figure 5.2: Calculation example for the deadline calibration

Figure 5.2 shows an example of the query deadline calibration with the QEP of Figure 5.1, setting b=2 for the passive backups of the Snapshot Builders and Computers and b_{cc}=1 for the active backups of the Computing Combiner. We can see that at level 0, the SB has a *MEPT* equal to δ, which corresponds to the emission time of the DCs' contributions, while the SB's backups have a *METB* equal to 0. Indeed, the execution time of the backups is only counted from their activation time, since the data replication is done in parallel with the primary node and the emission time is therefore already integrated. From this figure we can see that the calculation of the deadline is strongly influenced by the height of the tree and the number of backups. Note that the horizontal and vertical partitioning have no impact on the deadline calibration as long as they do not change the height of the QEP and the number of backups per partition.

5.1.2 Impact on Validity and Confidentiality

Satisfying the Validity property requires that the dataflow between the operators is consistent wrt. at least one of the $\zeta_Q(E)$ snapshots. However, with failure or message loss, a Snapshot Builder and its backups may build different snapshots, each belonging to $\zeta_Q(E)$. Therefore, three situations must

be distinguished to guarantee that the query result is equivalent to the one obtained with a centralized execution over at least one snapshot built by the Snapshot Builder or one of its backups. This particular snapshot is hereafter called the *reference* snapshot.

1. Without any partitioning, a single Snapshot Builder feeds a single Computer. Hence, a reference snapshot can be identified whatever the execution. It is either the snapshot built by the Snapshot Builder primary if there is no fault presumption, or the snapshot built by one of the activated backups otherwise.

2. Similarly, with horizontal partitioning, the reference snapshot is simply the union of each partition's snapshot (backup or primary).

3. With vertical partitioning however, the Snapshot Builder feeds several Computers, some of them potentially considering the primary snapshot and some others the backup ones in case of fault presumption. This leads to an inconsistency, i.e., a result built over a snapshot that does not belong to any snapshot of $\zeta_Q(E)$.

Figure 5.1 illustrates this third situation, where each Computer evaluates a different statistic but must consider a same snapshot belonging to $\zeta_Q(E)$. To solve this problem, several solutions can be envisioned but all of them incur a significant overhead. We sketch below two of them that address this issue in a different way:

- *Consensus*: Synchronize the snapshot between the primary Snapshot Builder and its backups thanks to a consensus protocol (Figure 5.3.a). [81] proposes an effective consensus protocol for OppNets that matches the Edgelet context, at the price of a distributed consensus. This consensus stage requires that the Snapshots Builders backups be active in order to agree on a representative reference snapshot. Indeed, if the collected data had to remain encrypted, it would become difficult, if not impossible, to verify the representativeness predicates.

- *QEP restructuring*: Rearrange the QEP so that the parallel branches are serialized, one after the other (Figure 5.3.b), thus avoiding inconsistencies due to multiple successors, but at the price of losing QEP parallelism and exploding the recursive runtimes. This solution implies extending the deadline, otherwise the intermediate timeouts would be too short and almost all the backups would be triggered, which would harm the privacy of the contributors since their data would be exposed several times.

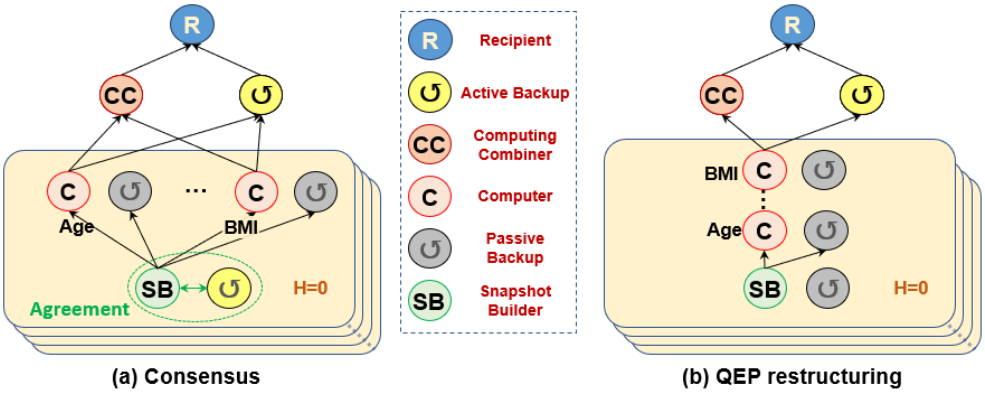

Figure 5.3: Solutions to guarantee the Validity property

Regarding the Confidentiality property, we notice that the risk of exposure of individual data is not the same whether the backups are passive or active. Indeed, as we explained previously, Passive Backups only decrypt data if they are activated, unlike Active Backups which decrypt them anyway. This protection is possible because, even if the edgelet is compromised, the integrity property of the code is still ensured (see Section 4.3) and guarantees that data can be decrypted only at the reception of an activation message from a successor node. Moreover, we consider that the activation messages are signed with the private key of the successor nodes, making them impossible to forge. Hence, besides horizontal and vertical partitioning, Passive Backups provide an additional layer of protection against confidentiality attacks on edgelets. Their use on all QEP operators is however not always possible, especially with the Computing Combiner which must be actively replicated, and with the Snapshot Builders when the QEP integrates vertical partitioning and a consensus protocol, as explained above.

5.2 Overcollection-Based Execution Strategy

This section introduces a very different way to handle the problem of Resiliency, which integrates the OppNet context by design. Contrary to the backup-based strategy, messages delays or loss are no longer considered as faults that must be recovered but rather as a legitimate behavior.

5.2.1 Enforcing Resiliency

As an alternative to securing every SPF in a QEP thanks to backups, we suggest over-collecting the dataset of interest so that the QEP may survive the loss of parts of it. To explain the intuition, let us consider a query over a sample dataset (e.g., *2000* individuals with *age>65*) where Data Processors (i.e., Snapshot Builder and Computers) execute distributive operators. Instead

of executing the operators on single edgelets, we distribute (using hashing) its execution over *n+m* edgelets where each one processes a partition of the original dataset, with *n* the minimum number of partitions to be collected and *m* the overcollection parameter (see Figure 5.4).

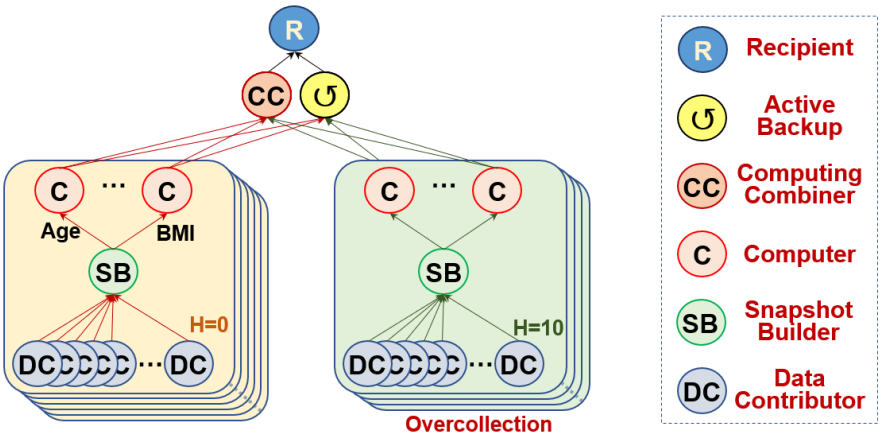

Figure 5.4: Resiliency based on the Overcollection strategy

The Overcollection ratio must be adapted to the presumed fault probability p_f of the OppNet to reach the expected success rate p_s for a query. Given that the probability of success of one partition is $p_p = (1 - p_f)^{|SPF|}$, with $|SPF|$ the number of SPFs in each partition, the number *m* of additional partitions is determined by the inequality $\sum_{i=n}^{n+m} \binom{n+m}{i} \times {p_p}^i \times (1 - p_p)^{(n+m-i)} \geq p_s$, i.e., at least *n* of the *n+m* partitions must succeed. Concerning the Computing Combiner, we assign Active Backups exactly like those of the Backup strategy.

5.2.2 Impact on Validity and Confidentiality

If all QEPs can satisfy the Validity property when executed in a backup mode (in some cases by adding a consensus among backups), this is no longer true when executed in an Overcollection mode. Indeed, (1) the complete QEP must be reorganized to handle a partitioned dataset[3] and (2) a reference snapshot $D \in \zeta_Q(E)$ must remain identifiable despite arbitrary loss of subparts of this dataset during the processing. To tackle point (1), a brute-force solution is to reorganize the QEP in a set of sub-QEPs, each performing an independent processing over a partition of the collected dataset with the Computing Combiner assembling the final result (see Figure 5.4). This solution applies only if the commutativity rules between operators allow to push all distributive operators down to the sub-QEPs and to push all non-

[3] In fact, the strategy remains applicable for a single partition but that would be equivalent to duplicating the QEP.

distributive operators up to the Computing Combiner. A QEP satisfying this condition is said *reshapable*. Under this assumption, point (2) can be easily tackled. The reference snapshot D is simply the union of all partitions that contributed to the QEP computation up to the Computing Combiner. Assuming that each of the $n+m$ partitions locally satisfies the set of representativeness predicates P and have a cardinality $|D|/n$, the Validity property is trivially preserved as long as less than m partitions are lost.

This shift in the Resiliency strategy has also implications in the way individual data is exposed. On the one hand, the fact that there are no backups ensures by design that individual data is only exposed once, as only the primary nodes will be able to decrypt the data and, in case of compromise, the data leakage is restricted to the compromised edgelets (see Section 4.3). On the other hand, the Overcollection mechanism implies that more Data Contributors must be involved in the query to ensure the success of the execution. Therefore, the disadvantage of the individual exposure in the Backup strategy is replaced by a higher collective exposure.

5.2.3 Relaxing Validity

Most data intensive queries of interest in our context are distributive by nature (as confirmed by various MapReduce or Spark implementations). However, some of the corresponding QEPs cannot be reshaped following the Brute-Force approach and then cannot combine Overcollection and Validity. This is notably the case of general interest machine learning algorithms, because they are iterative or need to exchange partial results computed over different data partitions. In these cases, a reference snapshot $D \in \zeta_Q(E)$ can no longer be identified in case of messages loss (e.g., two iterations may consider a different snapshot state). On the other hand, strict Validity is not a prerequisite for these algorithms which usually produce an approximate result. We thus suggest another basic preliminary method to handle these cases, called Iterative Brute-Force and sketched in Figure 5.5.

To execute an algorithm $\mathcal{A}$ with Iterative Brute-Force, each edgelet implementing a sub-QEP Computer iterates on (1) a *local convergence* phase where it computes $\mathcal{A}$ on its local partition and improves its local knowledge, initialized by a parameter of the sub-QEP, and broadcasts this knowledge to all others sub-QEPs, and (2) a *synchronization phase* where it receives the knowledge of the other sub-QEPs it has heard of and integrates them in its own knowledge. Right before the query deadline, the knowledge is sent to the Computing Combiner which combines all received knowledges and sends the final result to the Recipient.

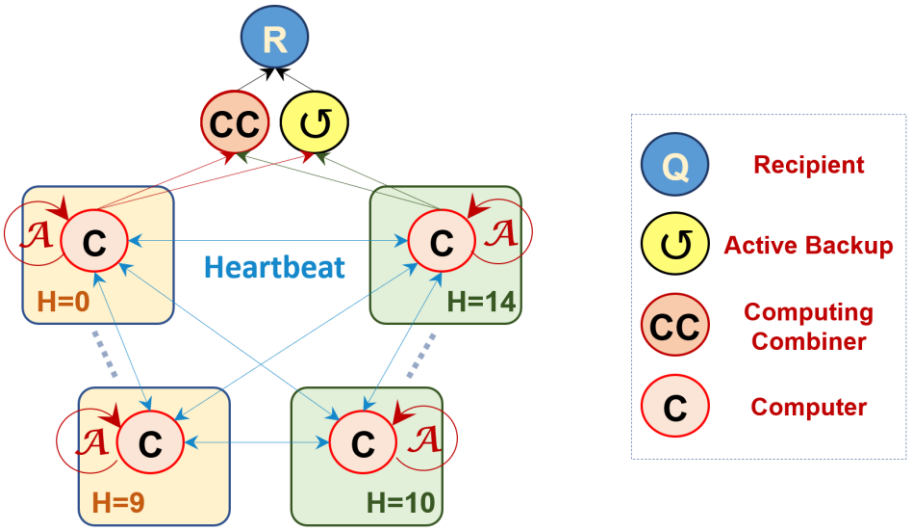

Figure 5.5: Overcollection with the Iterative Brute-Force method

The main question is when to stop the processing. Fixing a number of iterations a priori (with a minimal number of received messages) has little sense in the OppNet context where message delays, then edgelet progression, are unpredictable. Expecting a local convergence is also hazardous due to the instability of the synchronization phase among sub-QEPs. For instance, two edgelets with fast communications could converge locally quickly (without having even received any message from others) and decide to end prematurely their computation. Thus, we enforce the progression of the algorithm on all edgelets thanks to a *Heartbeat*, that is each iteration is cadenced by a clock, whatever the local state of the processing (i.e., a Computer moves to the next iteration even if few or no messages were received). Finally, local result is delivered when the deadline is imminent.

```
Iterative Brute-Force
Computer Edgelet (local_partition, initial_knowledge)
  knowledge ← initial_knowledge
  Heartbeat until (query_deadline - 1 round)
    Local conv: knowledge ← 𝒜(local_partition)
                and broadcast knowledge to all
    Synchro: knowledge ← received knowledge of others
  Send final knowledge to Computing Combiner
Computing Combiner
  combine all received knowledge and send to Recipient
```

We illustrate this method on three classical algorithms, namely Apriori, K-means and Stochastic Gradient Descent (SGD), and show its effectiveness in terms of proximity of results wrt. centralized executions in Chapter 6. While the first two algorithms are quite specific in the tasks to be performed, the

SGD algorithm is used to solve many well-known machine learning models in the literature [166]. Its relevance in the Edgelet context thus reinforces the applicative potential of the approach.

Apriori [167]: mine frequent itemsets to learn association rules.

- *knowledge*: frequent itemsets and their support (initially empty).
- *Local convergence*: $\mathcal{A}$ first computes the local support of all frequent itemsets in its partition then iteratively computes the local support of itemsets that are frequent in other sub-QEPs it has heard of.
- *Synchronization*: adds frequent itemsets of others in knowledge.
- *Computing Combiner*: sums the local supports of the common itemsets found in all received knowledge.

K-means [168]: form k clusters minimizing the intra-cluster variance.

- *knowledge*: current centroids (initially, k initial centroids)
- *Local convergence*: until local convergence or heartbeat, $\mathcal{A}$ assigns each element of its own partition to the cluster having the nearest centroid and recomputes the centroids of the new clusters, updating its knowledge.
- *Synchronization*: computes, on a cluster basis, the barycenter of all centroids received from other sub-QEPs, and integrates the result in knowledge.
- *Computing Combiner*: computes, the barycenter of all centroids received.

SGD [169]: adjust the weights of a model to minimize its objective function.

- *knowledge*: vector of weights *w* (initially random).
- *Local convergence*: until local convergence or heartbeat, $\mathcal{A}$ computes the gradients associated with the *w* vector for a small sample of its data (mini-batch) in order to iteratively update the model.
- *Synchronization*: computes the average of the all *w* vectors received from other sub-QEPs it has heard of, and integrates the result in knowledge.
- *Computing Combiner*: computes the average of the all *w* vectors received.

5.3 HYBRID-BASED EXECUTION STRATEGY

As the name suggests, the Hybrid strategy proposes a new method that combines the previous mechanisms, namely Backup and Overcollection, with the advantage of being able to adapt and benefit from each method depending on the situation.

5.3.1 Enforcing Resiliency

We have seen that the Backup strategy causes inconsistency problems when the QEP includes vertical partitioning. To avoid these problems, the idea of the Hybrid method is to assign backups only to Computers and use the Overcollection principle to compensate for the probability of fault presumption of the Snapshot Builders (see Figure 5.6). In fact, integrating backups within partitions is a smarter resiliency solution than continuously adding overcollection partitions, especially when the degree of vertical partitioning is high (since each SPF must survive for the partition to succeed). In this configuration, the probability of success of one partition is determined by the equation $p_p = (1 - p_f) \times (1 - p_f^{(1+b)})^{(|SPF|-1)}$, with p_f the probability of fault presumption, $|SPF|$ the number of SPFs in the partition (i.e., Snapshot Builder and Computers) and b the number of backups per Computer. As with the Overcollection strategy, the success for the $n+m$ partitions is constrained by the inequality $\sum_{i=n}^{n+m} \binom{n+m}{i} {p_p}^i (1 - p_p)^{(n+m-i)} \geq p_s$, with p_s the expected probability of success. Once again, we use Active Backups to make the Computing Combiner fault tolerant.

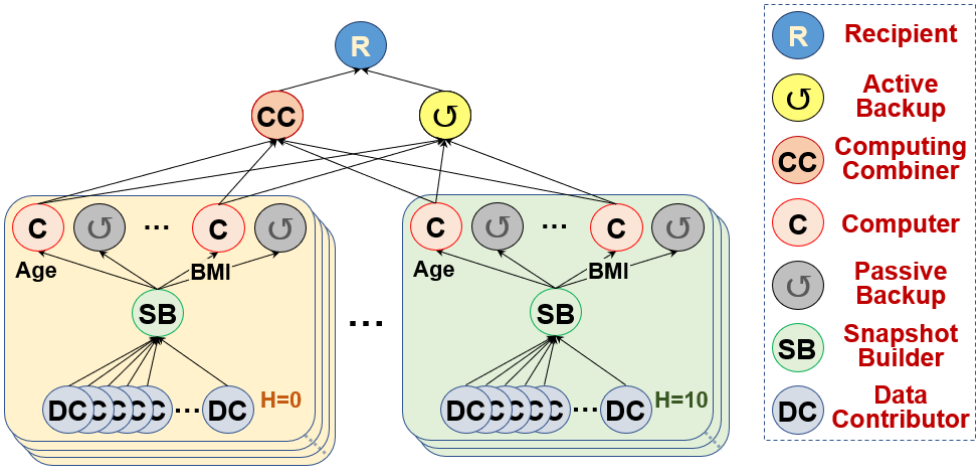

Figure 5.6: Resiliency based on the Hybrid strategy

Note that both parameters b and m can be adjusted to achieve the success rate p_s, thus we can design different optimization strategies depending on whether it is better to have more backups or more overcollected partitions. For example, we can choose them to minimize the number of nodes involved in the QEP or to minimize the number of messages sent over the network.

5.3.2 Impact on Validity and Confidentiality

Similar to the "pure" Overcollection strategy, the Hybrid execution plans use only one Snapshot Builder per partition. This mechanism ensures by construction that the snapshot is unique for each partition. Thus, even if the QEP incorporates vertical partitioning, the Computers are guaranteed to be consistent because they process exactly the same snapshot. However, the Validity is not so trivially guaranteed when the queries are iterative and the Computers of multiple partitions have to exchange information before producing their final result. Indeed, as explained in the previous section, the strict Validity property would require that all Computers in all partitions exchange the same information (data, aggregates or hints) at each iteration to form a consistent view. But if we follow the same approach as before, i.e., relaxing the property to tolerate partial and non-uniform exchanges between Computers, we then lose the interest of having added backups to the Computers since each Computer will only need to receive information from a subset of the others, and therefore there is no necessity to activate the backups. This particular issue will be further detailed in the next section.

Avoiding data inconsistency problems is not the only advantage of the Hybrid solution. While Active Backups on Snapshot Builders may be necessary to ensure data consistency in the Backup strategy (consensus solution), this is no longer the case in Hybrid since each partition has only one non-backuped Snapshot Builder. This limits the exposure of individual data involved in the representativeness predicates to a single node instead of $1 + b$ for the Backup strategy. Moreover, the mix between backup and overcollection mechanisms allows for a trade-off between individual and collective exposure. Although it is difficult to say which exposure is best, as it depends on the context, we will see in Section 6.1 that this exposure is often an order of magnitude lower than Backup alone or Overcollection alone.

5.4 QUALITATIVE EVALUATIONS

This section compares qualitatively the three execution strategies. Our goal is to guide a potential Recipient towards the right execution model when designing a computation dedicated to the Edgelet computing paradigm. This

choice depends on the type of computation to be performed, the way to define a representative snapshot for this computation and the expected query deadline. We can draw some design rules from the statements summarized in Table 5.1.

<table>
<tr><th>Resiliency</th><th>Type of computation</th><th>Snapshot definition</th><th>Validity</th><th>Confidentiality</th><th>Success Rate</th></tr>
<tr><td rowspan="4">Backup based resiliency</td><td>Horizontally partitionable</td><td>P partitionable</td><td>By construction</td><td rowspan="3">Activated backup exposed</td><td rowspan="3">Requires large query deadline</td></tr>
<tr><td>Vertically partitionable</td><td>Any P</td><td>Consensus</td></tr>
<tr><td>Others</td><td>Any P</td><td>By construction</td></tr>
<tr><td colspan="3">Iterative</td><td colspan="2">Unrealistic</td></tr>
<tr><td rowspan="3">Over collection based resiliency</td><td>QEP reshapable</td><td>P partitionable</td><td>By construction</td><td rowspan="2">Over collected data exposed</td><td rowspan="2">Supports very small query deadline</td></tr>
<tr><td>Iterative reshapable</td><td>P partitionable</td><td>Ground validity</td></tr>
<tr><td colspan="3">P not partitionable</td><td colspan="2">Invalid</td></tr>
<tr><td rowspan="2">Hybrid based resiliency</td><td>QEP reshapable</td><td>P partitionable</td><td>By construction</td><td>Activated backup and over collected data exposed</td><td>Supports small query deadline</td></tr>
<tr><td colspan="3">Iterative or P not partitionable</td><td colspan="2">Inappropriate</td></tr>
</table>

Table 5.1: A taxonomy of execution strategies

In this table, *Horizontally/Vertically partitionable* refers to the property of the computation to be distributed among several Data Processors, as explained in Section 4.3. Both forms of partitioning greatly make sense when conceiving a computation dedicated to the Edgelet computing paradigm, either to minimize the amount of data exposed at each Data Processor or to avoid the exposure in the same Data Processor of information sensitive when combined or even to minimize the Data Processor workload when energy consumption matters. While vertical partitioning does not impose additional constraint on the processing, horizontal partitioning requires that the set of representativeness predicates P be itself partitionable, i.e., that it can be applied to each partition independently (e.g., *age* > 65 is partitionable while median(*age*) < 10 is not).

QEP reshapable refers to the capacity to reorganize distributive and non-distributive operators in the QEP to be computed. This is a prerequisite to

exploit the Overcollection-based strategy for this QEP. *Ground validity* means that the Validity property defined in Section 3.5 cannot be enforced; hence, it is up to the Recipient to assess empirically the accuracy of the final result, as we did in Section 6.3 for Apriori, K-means and SGD. Finally, the *Confidentiality* column expresses the additional amount of data exposed by each strategy compared to an ideal strategy without resiliency (i.e., without backups nor overcollection).

Based on this table, we can draw the following conclusions. First, if the QEP implementing the computation cannot be reshaped or if P is not partitionable, the Backup strategy is the only solution since applying overcollection would be equivalent to completely duplicating the query. Second, if the computation is iterative, the Overcollection strategy turns to be the only solution as well. Indeed, using backups, a consensus (or an equivalent mechanism) would be required at each iteration to ascertain that all participants consider *in fine* the same reference snapshot. Otherwise, only a Ground validity can be expected, but Overcollection outperforms Backup and Hybrid in this case (no additional protocols related to interactions with backups). Third, if the QEP is reshapable and P is partitionable, the right choice between Backup, Overcollection and Hybrid is driven by the expected success rate and by privacy considerations. Using Backup or Hybrid strategies, the query deadline must be calibrated to accommodate the number of backups defined at each QEP level (i.e., activate them one after the other in case of fault presumption), a factor which disappears with Overcollection alone for which the query deadline depends only on the average network latency and the number of levels in the QEP. This explains the query deadline requirements, which range from large for the Backup strategy to very small for the Overcollection strategy.

Regarding privacy, the three methods do not expose data in the same way. Indeed, as explained in the previous sections, TEEs guarantee that data at rest is not exposed in backups until they are activated. Hence, the same personal data is potentially exposed in as many backups as required by the satisfaction of the Resiliency property and this number increases with the presumed fault rate of the OppNet. With Overcollection alone, in contrast, the same data is never exposed twice. However, data from a larger population of individuals must be involved in the computation due to overcollection. The Hybrid strategy combines both types of exposure, with the advantage of being able to adjust the proportion of each one by calibrating the number of backups and the number of overcollected partitions. This choice made by the Recipient as to how to expose personal data may influence the approval of the Regulator as well as the consent of the participants.

The taxonomy of solutions presented in Table 5.1 is still preliminary and more subtle design rules can be envisioned. In particular, the Ground validity should be more deeply investigated with the goal to identify finer classes of algorithms for which better validity guarantees can be expected. For instance, the Apriori implementation sketched in Section 5.2.3 exhibits the salient feature that Validity can be assessed *a posteriori* by the Computing Combiner. Indeed, (1) the reference dataset is the union of the partitions it received from the Data Processor it has heard of, and (2) a frequent itemset is necessarily frequent in at least one of these partitions. The Computing Combiner must simply check that enough information has been received to compute the support of all these candidate frequent itemsets. We expect also be able to guarantee the convergence for some algorithms when specific conditions are met but let these issues for future work.

6 VALIDATION

This chapter presents our quantitative evaluations of the execution strategies presented earlier. Our objective is to validate the relevance of the Edgelet approach, calibrate the system parameters and verify its effectiveness.

We first compare the three execution strategies, namely Backup (Bak), Overcollection (Ovr) and Hybrid (Hyb), providing insights to properly configure the resiliency parameters. Then, we implement a non-iterative Edgelet execution with the aim of calibrating the query deadline and achieving the targeted success rate. Finally, we test the iterative methods Apriori, K-means and Stochastic Gradient Descent to evaluate the quality of their results against a centralized execution.

6.1 COMPARISON OF EXECUTION STRATEGIES

In this first section, we want to study the respective behavior of the execution strategies presented in Chapter 5. To enable a fair comparison between the three resiliency mechanisms, we assume that the query deadline is correctly calibrated so that each message sent in the OppNet has enough time to reach its destination (see Section 6.2 for deadline adjustment). Consequently, the probability of fault presumption p_f is reduced to the probability of device failure. Our goal is to observe the consequences of the resiliency methods, particularly in terms of personal data exposure and network overload.

6.1.1 Overall Analysis

To begin with, we examine the general impact of execution strategies on the Query Execution Plans (QEPs). We want to observe the transformations induced by the resiliency mechanisms (Bak, Ovr and Hyb), and to do so, we count the number of additional nodes introduced in the QEPs. Since these are redundant nodes, their presence in large numbers will be seen as a disadvantage, as they lead to additional data exposure and network overload.

	Passive Nodes	Active Nodes
Bak	if (|C|=1): $\|SPF\| \times b \times n$ else: $\|C\| \times b \times n$	if (|C|=1): 0 else: $b \times n$
Ovr	0	$\|SPF\| \times m$
Hyb	$\|C\| \times b \times (n+m)$	$\|SPF\| \times m$

Table 6.1: Formulas for the Additional Nodes

Table 6.1 shows the formulas used to determine the number of additional nodes for each strategy. We distinguish two types of nodes: those that are active, working in parallel with the primary nodes (active backups and nodes in overcollected partitions); and those that are passive, waiting to be activated (passive backups). Concerning the notations, we note |*SPF*| the number of Single Point of Failure in each partition, i.e., 1 Snapshot Builder and |C| Computers. For Bak, the number of backups b is calibrated using the formula of Section 5.1.1. Note that when |C|>1, we consider that the Snapshot Builders' backups must be active to establish a consensus on the dataset (see Section 5.1.2). For Ovr, the number of additional partitions m is adjusted using the formula of Section 5.2.1. And for Hyb, the pair (b, m) is chosen using the formula of Section 5.3.1 in order to optimize the sum of the number of additional nodes (passive and active).

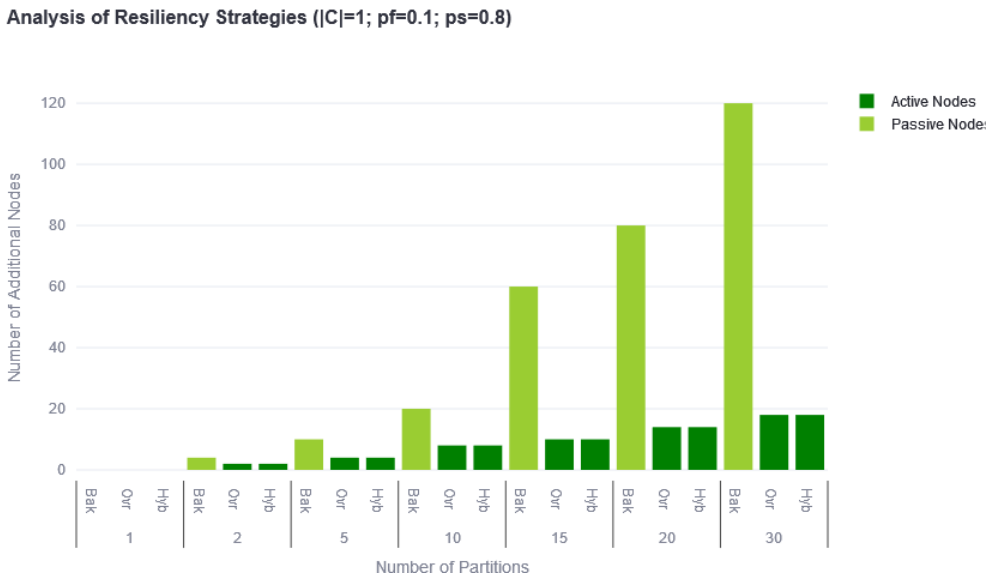

Figure 6.1: Additional Nodes per number of partitions

Figure 6.1 presents the number of additional nodes for each strategy depending on the number of partitions (horizontal partitioning). We first study a basic setup with a single Computer in each partition (|C|=1) and a device failure probability of 10% (p_f=0.1) for a success probability of 80% (p_s=0.8). We observe that increasing the number of partitions greatly favors

strategies exploiting the overcollection principle (Ovr and Hyb). Moreover, Hyb perfectly mimics Ovr by increasing the number of additional partitions *m* while maintaining the number of backups *b* equal to zero. Note that even if the additional nodes are only passive for the Bak strategy, we saw in Chapter 5 that they necessarily induce extra overhead. Indeed, they generate a lot of communication for data replication, hence the objective to minimize their number.

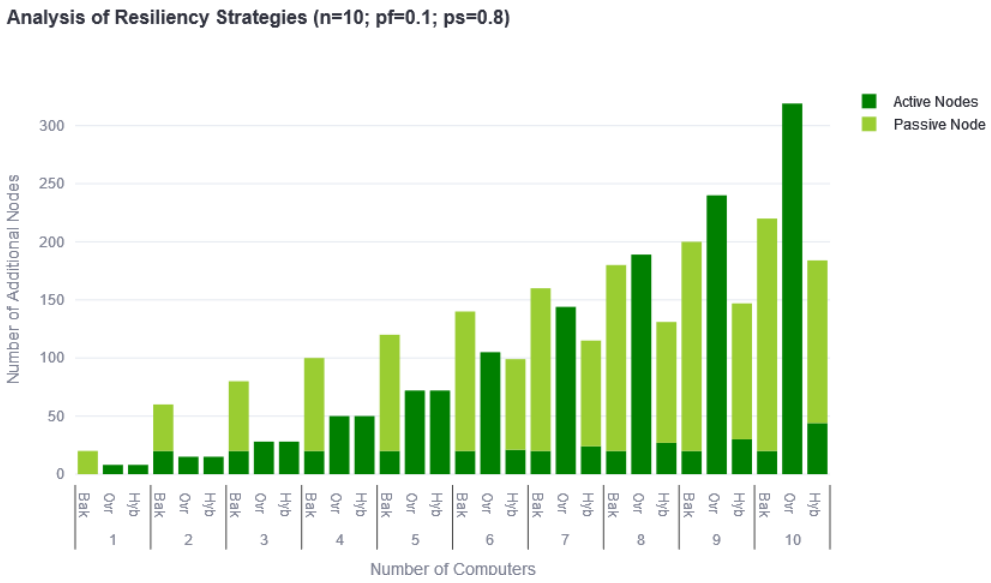

Figure 6.2: Additional Nodes per number of Computers (p_f =0.1)

Now that we have varied the horizontal partitioning, let us focus on the vertical partitioning, taking the same configuration as in previous chapters, i.e. n=10. Figure 6.2 shows that when the number of Computers per partition is low (|C|<6), Ovr and Hyb are identical and outperform Bak. From |C|≥6, we observe that Hyb begins to distinguish itself by finding a compromise between *b* and *m* that enables it to perform strictly better that the other two strategies. From |C|≥8, we see an inversion between the Bak and Ovr plots, with the number of additional nodes shifting in favor of Bak. As shown in Figure 6.3, this inversion occurs earlier as the probability of failure increase.

The explosion in the number of additional nodes for Ovr is explained by the fact that, for the strategy to succeed, at least *n* partitions with all SPFs must "survive" failures, whereas for Bak, only one primary or backup Data Processor per SPF (Snapshot Builder and Computers) is needed in each partition to guarantee successful executions.

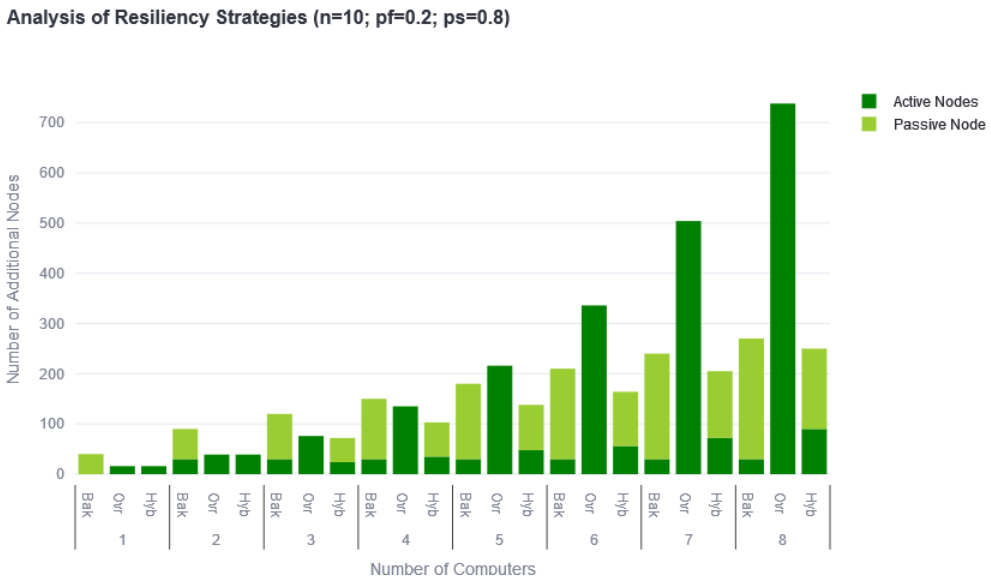

Figure 6.3: Additional Nodes per number of Computers (p_f=0.2)

Based on these results, we can draw the following conclusions: (1) horizontal partitioning greatly favors strategies based on overcollection (Ovr and Hyb), (2) vertical partitioning is detrimental to the Ovr strategy, especially when the probability of failure increases, and (3) the Hyb strategy is able to outperform Bak and Ovr by leveraging the strengths of both strategies.

6.1.2 Personal Data Exposure

Let us now consider the influence of these resiliency mechanisms on the exposure of personal data, both individually (i.e., how often the same data is exposed) and collectively (i.e., how many additional contributing individuals are included in the QEP).

	Individual Exposure	Collective Exposure
Bak	if (|C|=1): min=0, max=2 × b else: min=b, max=2 × b	0
Ovr	0	m/n
Hyb	min=0, max=b	m/n

Table 6.2: Formulas for the Additional Exposure

Table 6.2 gives the formulas used to calculate the additional exposure. Individual exposure indicates the number of times the same personal data is exposed. The minimum is determined by the number of active backups in the QEP, while the maximum is determined by the number of passive backups. Note that we count two additional exposures for Snapshots Builders and Computers backups, but that the number of Computers |C| is not taken into

account, as they process data of different individuals (otherwise, increasing vertical partitioning would undermine privacy). Regarding collective exposure, we consider the proportion of additional individuals involved in the QEP, which is given by the *m*/*n* ratio. Note also that, although these two measures aim to quantify additional exposure, they don't have the same meaning which makes them very difficult to compare. Individual exposure reveals the risk incurred by each individual, while collective exposure indicates the total number of individuals at risk. Depending on the application scenario and the type of queries, it will be the responsibility of the Recipient and the Regulator to find the right compromise.

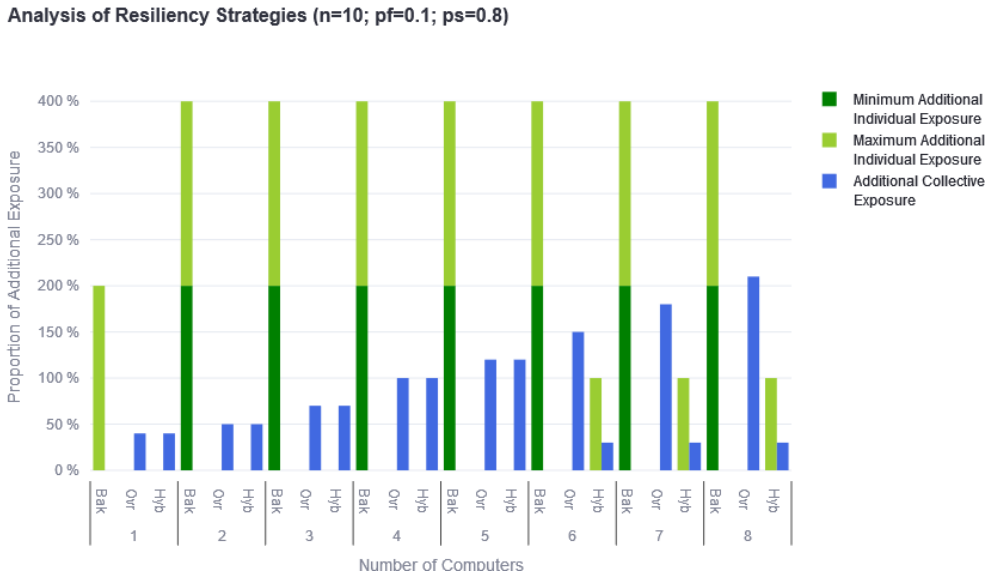

Figure 6.4: Additional Exposure per number of Computers (p_f=0.1)

Figure 6.4 shows on the same graph both types of exposure as a function of the number of Computers |C|. Although |C| is not directly considered when calculating personal data exposure, we saw in the last section that it strongly influences the number of backups *b* and the number of additional partitions *m*, which therefore impacts it indirectly. Let us take a look at what happens at |C|=2. For Bak, we observe that the minimal additional individual exposure is at 200%, indicating that the data is exposed to 2 active backups (corresponding to Snapshot Builders), and that the maximal additional individual exposure is at 400%, meaning that the data may also be exposed to 2 passive backups (those belonging to Computers). For Ovr and Hyb, we see an additional exposure at 50%, meaning that on the *n*=10 initial partitions, it is necessary to add *m*=5 other partitions to reach the target success rate of 80%.

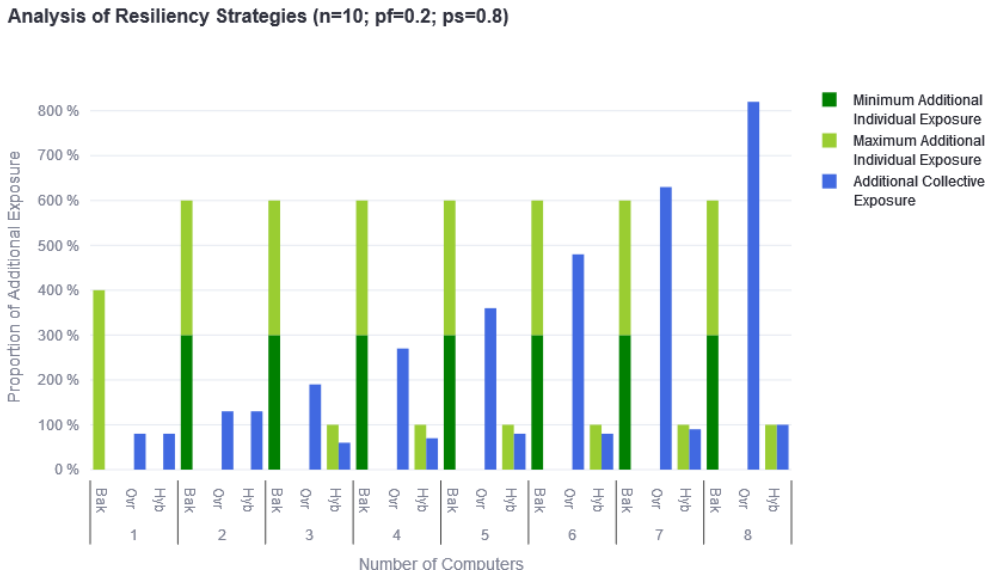

Figure 6.5: Additional Exposure per number of Computers (p_f=0.2)

Similar to the analysis in the previous section, we see that, as the number of Computers |C| increases, the Ovr strategy deteriorates. For example, when |C|=7 and p_f=0.2, we obtain an overcollection rate of over 600% (see Figure 6.5). For Hyb, we observe a mixture of the two exposures. Indeed, above a certain threshold, the Hybrid method simultaneously integrates passive backups and additional partitions, with the (*b*,*m*) pair again chosen to reduce the number of nodes in the QEP. In fact, optimizing these parameters according to personal data exposure is rather complicated and application dependent, as the two measures (individual and collective) are not comparable. Nevertheless, we can see that this strategy seems to be a good compromise when the other two are no longer acceptable (e.g., |C|=8 on Figure 6.4: one additional backup and 30% additional collective exposure).

6.1.3 Network Overload

To complete this analysis, we propose an evaluation of network overload by counting the number of additional messages generated by each execution strategy. Although our objective is not to optimize network communications (see Chapter 3), we nevertheless wish to study the impact of strategies on the congestion they may generate. To this end, we are interested in measuring the number of messages containing data sent over the network, ignoring any other type of message (e.g. backup activation messages). As message size depends on the use case and the type of query, we assume a single size for all messages sent (whether data contribution or intermediate result).

	Mandatory Messages	Potential Messages
Bak	if (|C|=1): $(\|D\| + n) \times b$ else: $[\|D\| + (2 + b) \times \|C\| \times n] \times b$	if (|C|=1): $(1 + b + \|CC\|) \times b \times n$ else: $\|C\| \times b \times \|CC\| \times n$
Ovr	$[\|D\|/n + \|C\| \times (1 + \|CC\|)] \times m$	0
Hyb	$[\|D\|/n + \|C\| \times (1 + \|CC\|)] \times m +$ $\|C\| \times b \times (n + m)$	$\|C\| \times b \times \|CC\| \times (n + m)$

Table 6.3: Formulas for the Additional Messages

Table 6.3 provides formulas for counting the number of additional messages generated by the three resiliency strategies (Bak, Ovr, Hyb). We observe two categories of messages: mandatory messages, which are necessarily sent when the QEP is executed, and potential messages, which are sent when passive backups are activated. Regarding notations, we note |*D*| the cardinality of the dataset required for the query and |CC| the number of Computing Combiners with its specific backups (|CC|=1+b_{cc}). Note that the number of messages sent by Data Contributors can be much higher than the target |*D*|. Indeed, snapshot construction requires an unpredictable number of contributions in order to satisfy the set of representativeness predicates *P*. As this unpredictable quantity of messages is context-dependent, we do not include it in our measurements.

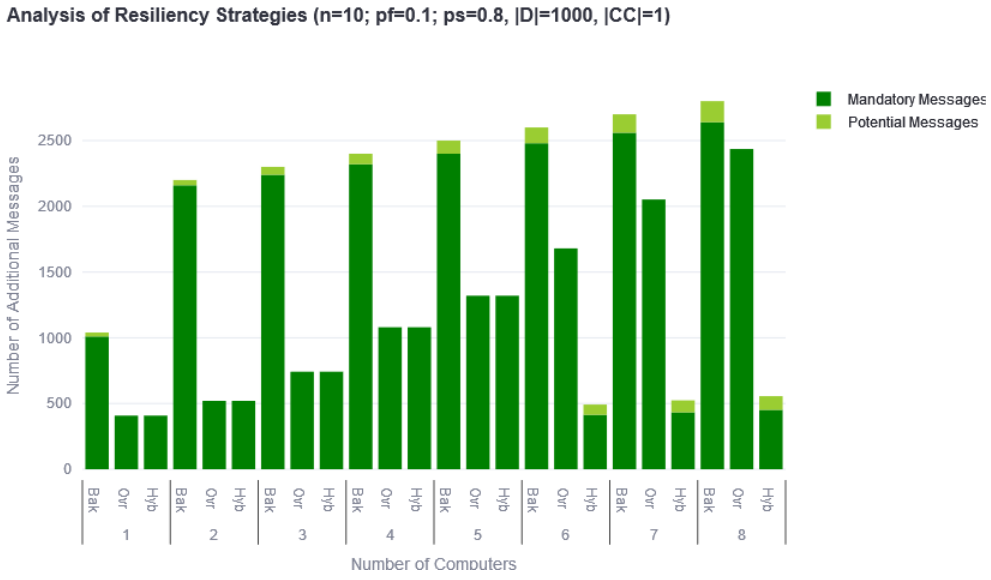

Figure 6.6: Additional Messages per number of Computers

Figure 6.6 shows the number of additional messages according to the number of Computers. We have calibrated a horizontal partitioning of *n*=10, a failure probability of 10% and a success rate of 80%. For these parameters, the QEP needs only one Computing Combiner, i.e. no active backups associated (b_{cc}=0). The cardinality of the target dataset is set to |*D*|=1000. As

might be expected, the vast majority of additional messages are related to the contribution phase. Bak is therefore disadvantaged, as every message sent by Data Contributors is replicated on the Snapshot Builder's backups. As for Ovr and Hyb, the strategies are again identical as long as |C|<6. This observation, already made in Figure 6.2, is explained by the fact that Hyb is still optimized to minimize the additional nodes in the QEP.

Let us now look at Figure 6.7 to see how Hyb behaves when the pair (b,m) is calibrated to optimize the number of additional messages (mandatory and potential). We can see that Hyb is systematically better than the other strategies, and that the gap widens as |C| increases. In fact, like Ovr, the Hybrid strategy is naturally adapted to minimize the number of messages sent during the contribution phase, since Snapshot Builders have no backups. The advantage of Hyb over Ovr is that it is able to compensate for the probability of partition failure by adding backups to the Computers. This feature proves to be a real asset, since reducing the number of additional partitions leads to a drastic reduction in the number of additional messages sent. Note that this optimization is not without cost, as it inevitably entails modifications in the exposure of personal data. For instance, with p_f=0.1 and |C|=3, Hyb will count 1 backup per Computer for a 20% overcollection instead of 0 backup for a 70% overcollection (see Figure 6.4).

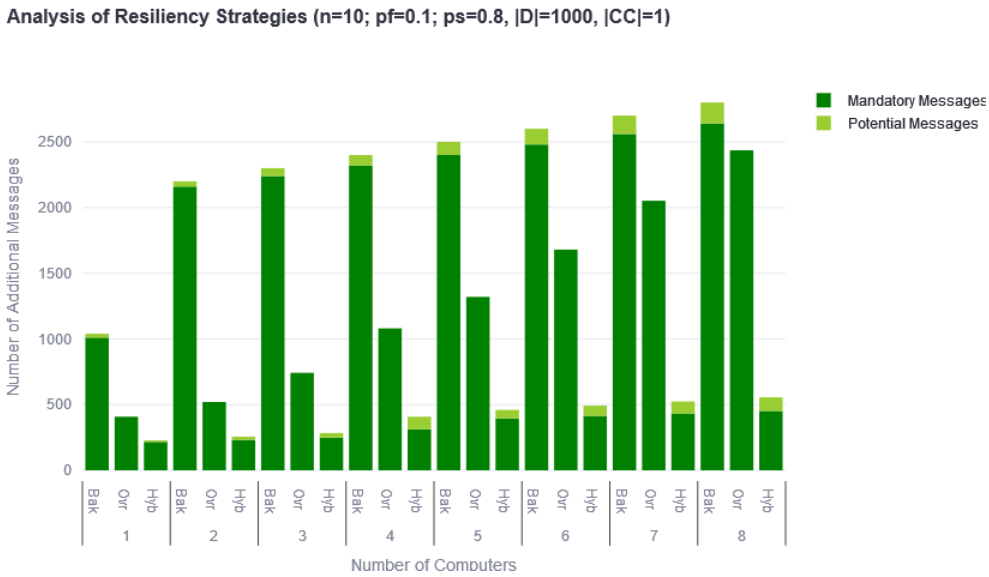

Figure 6.7: Additional Messages per number of Computers (Hyb optim)

6.2 ADJUSTMENT OF THE QUERY DEADLINE

In this second section, we focus on how to calibrate the query deadline such that messages sent in the OppNet have sufficient time to reach their destination. Without loss of generality, we will focus on implementing a non-iterative processing executed with the Ovr strategy. Indeed, adding backups (for Bak and Hyb) only shifts the deadline according to the procedure described in Section 5.1.1. Moreover, we have just seen in the previous section that the Ovr strategy is better in most configurations (with Hyb matching it by imitation). So, to study the query deadline, we built an Edgelet computing software on top of the Opportunistic Network Environment (ONE) simulator [77] providing detailed traces of OppNets communications (see Figure 6.8). We model two representative use cases with messages exchanged using an epidemic routing strategy [61], namely *Mall* and *DomY*.

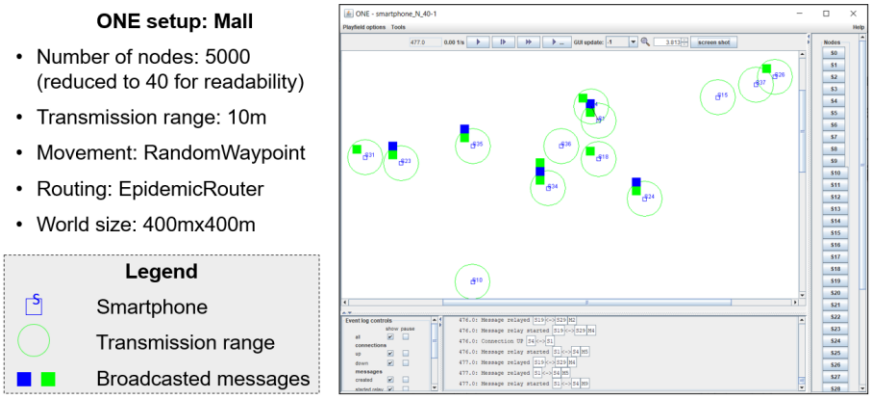

Figure 6.8: Mall simulation with the ONE

Mall: Edgelet computing within a Mall of 0.16 km2, where 5,000 edgelets (customer smartphones following a RandomWayPoint movement) are opportunistically executing a query exchanging messages using Bluetooth when meeting. The mean OppNet latency obtained with ONE is $\bar{L}$ = 1,936 s for a standard deviation σ = 933 s, resulting in a relative standard deviation of $R\sigma = \frac{\sigma}{\bar{L}} = 0.48$

DomY: the DomYcile project [21], with 8,000 personal home boxes (edgelets), 800 healthcare workers (with constrained routes in ONE) within the Yvelines district (2,284 km2). We obtained a mean latency $\bar{L}$ = 27,113 s with a relative standard deviation $R\sigma$ = 2.43, (i.e., σ = 65,794 s).

Using the latencies computed by ONE, we execute a non-iterative QEP following the Ovr strategy, considering vertical partitioning on 3 Computers with a device failure probability of 10%. We used 3 values of m/n, ranging

from 0.5 to 1.5. To enable comparisons between Mall and DomY, we used on x-axis the query deadline divided by $\bar{L}$, called α hereafter, and measure the query success ratio (y-axis on Figure 6.9). To minimize random variations, we averaged the results of 300 executions. Note that we are only interested in the impact of message transmission along the QEP and not in studying the quality of its processing, the latter being easily assured as long as fewer than *m* partitions are lost (see Section 5.2.2). Therefore, we consider an execution successful when at least *n* partitions transmit their response to the Recipient before the query deadline.

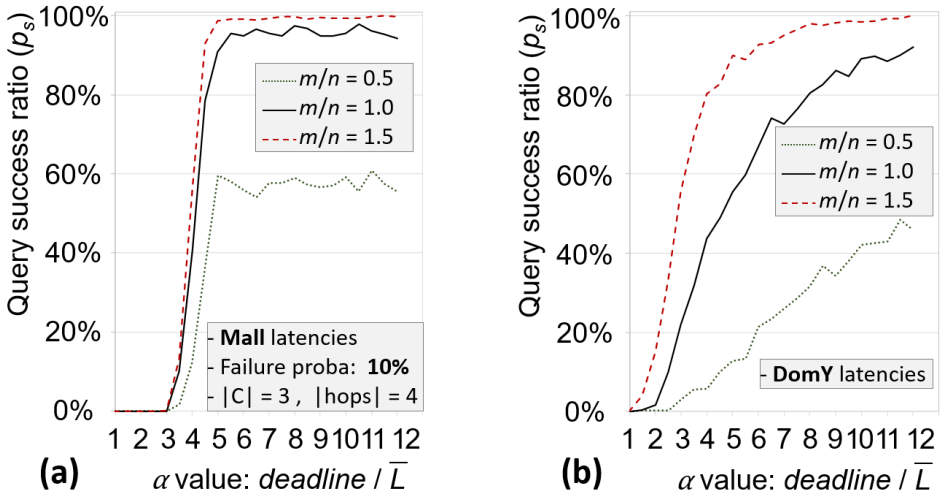

Figure 6.9: Query deadlines (for Overcollection)

From these plots, we make 3 observations: (1) in the Mall context (small *Rσ*), all executions complete successfully with $m/n \geq 1$ and $\alpha = 4$ (vertical increase); while the DomY context requires much larger deadlines due to its larger *Rσ* which impacts the fault presumption; (2) having an underestimated *m/n* value is risky: it reduces the ratio of successful queries and requires a significantly higher query deadline in contexts with large *Rσ* (e.g., DomY); (3) having a larger *m/n* value is rather useless for small *Rσ* contexts (e.g., Mall), indeed, a small *Rσ* means that latencies are close to the mean, thus a well calibrated *m/n* is sufficient to absorb few late messages.

In conclusion, the *m/n* ratio should not be underestimated and the query deadline should be fixed larger than $\bar{L}$×|hops| where |hops| is the number of hops in the query plan (in this case, 4: Data Contributor ➛ Snapshot Builder ➛ Computer ➛ Computing Combiner ➛ Recipient). Both *m/n* and the deadline should be overestimated when the OppNet latencies have a large *Rσ*. Thus, the query deadline should be fixed (for a 4 hops query) around 2 days for DomY and 2-3 hours for Mall, values that are quite reasonable given our application context.

6.3 QUALITY OF ITERATIVE COMPUTATIONS

In this third section, we examine the Edgelet iterative execution of the algorithms presented in Section 5.2.3. Our approach is empirical: we aim to assess the relevance of the Iterative Brute-Force method by verifying the quality of the results obtained compared with a centralized execution.

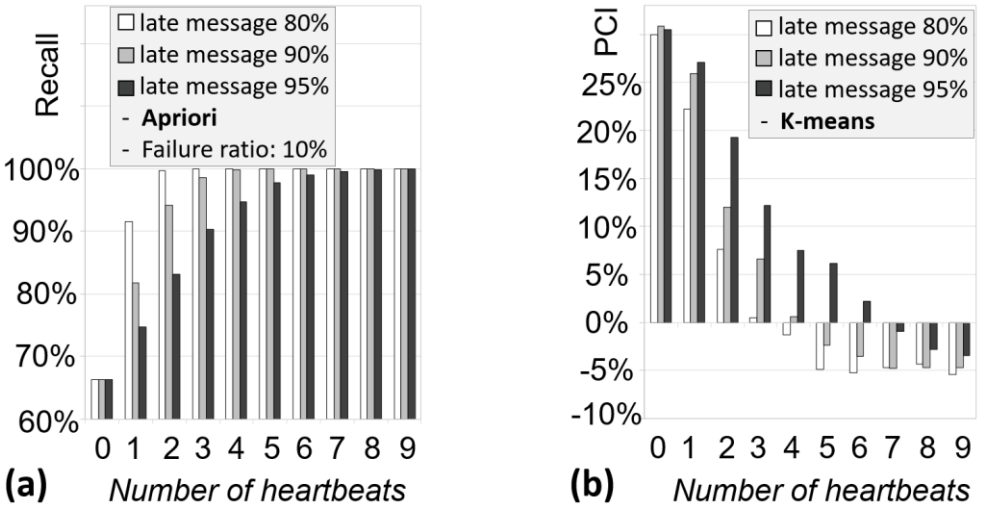

Figure 6.10: Heartbeat execution quality (Apriori and K-means)

We first simulate the iterative algorithms Apriori and K-means, considering synthetic and real data sets used to evaluate their quality (e.g., [170] for Apriori). These Edgelet executions consider the QEP of Figure 5.5 with a horizontal partitioning of n=10 for m=10 additional partitions. We systematically measure the quality of Edgelet executions against centralized fault-free executions (y-axis on Figure 6.10). More precisely, for Apriori, we compare the association rules generated after frequent item mining in Edgelet executions to those obtained in centralized ones, and use the precision/recall metrics to assess the comparison. Similarly, for K-means, we compute the Percentage Change Inertia (PCI), i.e., the percentage change between the Edgelet inertia (intra-cluster variance) and the centralized one. The x-axis indicates the number of heartbeats during the query. To evaluate iterative methods in extreme conditions, we reduced artificially the heartbeat duration such that the observed proportion of late messages (i.e., messages arrived after the Heartbeat) is 80%, 90% and 95% (see Figure 6.10).

We observe that even with no iteration, Apriori reaches 65% recall and K-means reaches 30% degradation of its inertia. Both converge quite quickly towards a recall of 100% or a PCI < 0% (4 or 5 heartbeats for 80 or 90% late messages, 7 or 8 with 95% late messages). Note that with Apriori, precision

is always 100% (not shown). Indeed, as we verify that the Computing Combiners always receive *n* or more sub-QEPs results, we can thus remove potential false positive. With K-means, we observe that as the number of heartbeats increases, the PCI can become negative. The reason for this is that an Edgelet calculation with many heartbeats can be better than a centralized calculation, thanks to the fact that it considers up to m additional partitions.

We then simulated the iterative Edgelet execution of Stochastic Gradient Descent (SGD) on a classification problem (Adult Income [171]) and a regression problem (California Housing [172]). As for Apriori and K-means, we consider the QEP of Figure 5.5 with a horizontal partitioning of n=10 for m=10 additional partitions. For classification, the underlying learning model used is a support vector machine, while for regression it is a linear model. For both, we use a constant learning rate and divide the number of maximum local iterations by a factor of 50 compared to centralized executions in order to reduce the amount of work on each edgelet. We then compare the accuracy of the model obtained on test data (mean accuracy for classification and coefficient of determination R^2 for regression).

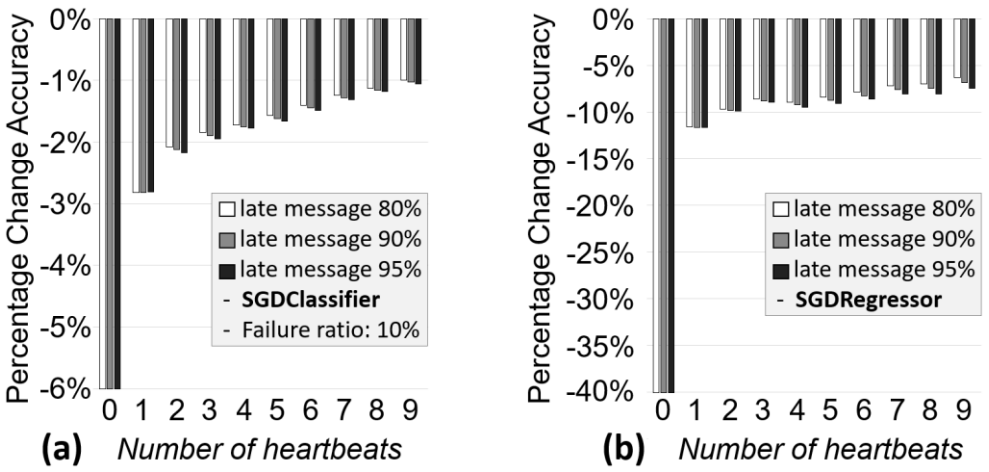

Figure 6.11: Heartbeat execution quality (SGD)

Figure 6.11.a and Figure 6.11.b show results of heartbeat executions on the two problems (classification and regression). The y-axis indicates the Percentage Change Accuracy (PCA), i.e. the percentage change between the model accuracy obtained with an Edgelet execution and that obtained with a centralized execution. At first glance, we can see that models built in Edgelet gradually converge towards the quality of centralized models. Even if the initial PCA is variable (-6% for classification and -40% for regression), since it depends on the learning rate and the maximum number of local iterations chosen, it does not prevent the models from progressing. Then, we

notice that the proportion of late messages is not a determining factor for convergence speed. In fact, the SGD algorithm's ability to work on small samples (mini-batches) at each iteration makes it a prime candidate for distributed and decentralized execution [169]. As a result, when the data is correctly distributed over the Computers, they compute a local model that naturally converges towards the best parameters.

6.4 CONCLUSION

Several lessons can be drawn from the quantitative evaluations carried out in this chapter:

Firstly, based on several metrics (additional nodes, exposure, network overload), we are able to compare very precisely our three execution strategies: Bak, Ovr and Hyb. We have observed that the more the execution plans are distributed (horizontal partitioning), the better the strategies based on overcollection (Ovr, Hyb) perform compared to the Bak strategy. Indeed, the overcollection mechanism enables very fine-tuning of resiliency to achieve the target success rate, rather than systematically adding backups to all QEP nodes. We also found that when the number of Computers per partition (vertical partitioning) is not high, the Ovr and Hyb strategies are strictly identical (Hyb imitating Ovr). Above a certain threshold, the Hyb strategy becomes more efficient, especially in terms of network overload.

Secondly, the implementation of OppNet use cases in the ONE simulator [77] has enabled us to better understand how to calibrate the query deadline. On the one hand, we found that it necessarily had to be adjusted according to the average latency of messages and the number of hops in the QEP, as a deadline chosen too short would inevitably lead to a very low success rate. On the other hand, we have seen that adding more partitions for Ovr (which also applies to Hyb) speeds up requests towards the desired success rate. In fact, for the same OppNet configuration, the higher the number of partitions, the greater the probability that some will be faster than others. Note that even if excessive overcollection can shorten the deadline, it comes at the price of greater collective exposure and network overload.

Thirdly, heartbeat executions show quite good results on Apriori, K-means and SGD despite an (artificially) high ratio of late messages. Indeed, we have found that, as iterations progress, these three algorithms converge towards the same accuracy as centralized executions. The results thus obtained indicate that machine learning algorithms can be performed in the Edgelet context while enforcing Confidentiality, Resiliency and Validity.

7 IMPLEMENTATION AND PRACTICAL USE CASES

In this chapter, we will show the implementation of the Edgelet computing paradigm through two demonstrations. The first one applies to the real use case DomYcile with messages sent in OppNets while the second one proposes to extend the usages with heterogeneous devices communicating in weakly connected settings. Although these applications share similarities, we will see that the objectives achieved are quite different, ranging from practical implementation to adaptation to other contexts.

7.1 MEDICAL USE CASE IN OPPNETS

This demonstration is a concrete application of the Edgelet computing paradigm to the DomYcile use case [21]. Our goal is to evaluate the relevance of the approach through complex distributed processing on sensitive data whose computations are performed on low-capacity and opportunistically connected personal devices. In this section, we will first give an overview of the implemented platform, then present the realized execution scenario and finally conclude on the obtained results.

7.1.1 Implemented Platform

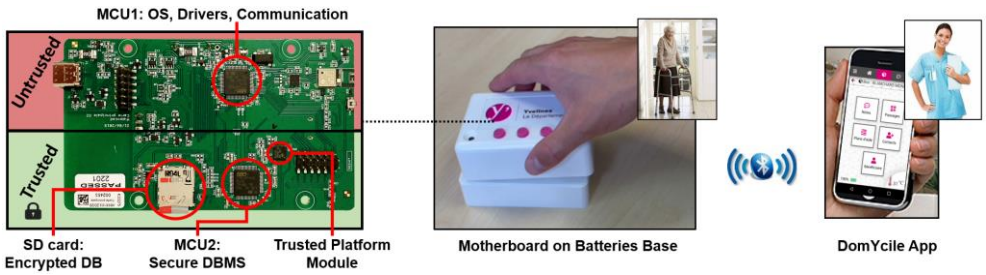

Figure 7.1: Hardware of DomYcile secure boxes

Hardware platform. It includes a set of personal secure boxes, as deployed and delivered to each patient in the DomYcile project. As illustrated in Figure 7.1, medical and social workers interact with patient folders hosted in the boxes through a smartphone application, thus completing their data over time. These secure boxes incorporate two STM32F417 microcontrollers (MCUs). The first MCU is dedicated to communications with the outside while the second manages the recorded data. To be even more precise, the second MCU is connected to a µ-SD card hosting the patient's raw data, and to a tamper-proof TPM (Trusted Platform Module). This TPM secures the cryptographic keys and guarantees, during the secure boot, that the embedded code has not been tampered with. Hence, these secure boxes act as TEE-enabled devices and play the role of edgelets.

Software platform. As shown in Figure 7.2, the demo software platform consists of the following components: (1) a Graphical User Interface (GUI) implemented in Dash Python [173] that allows interactive configuration and visualization of Edgelet queries; (2) an Edgelet manager that orchestrates executions and communications between edgelets; (3) an OppNet modeler that models the massive distribution of edgelets over the city of Versailles[1], chief town of the Yvelines district. The OppNet modeler itself relies on the ONE simulator [77] and uses the generated message traces for the Edgelet manager to coordinate the executions in a similar way. Note that the Edgelet manager and OppNet modeler are only necessary for the demonstration and are not part of the platform deployed in the field.

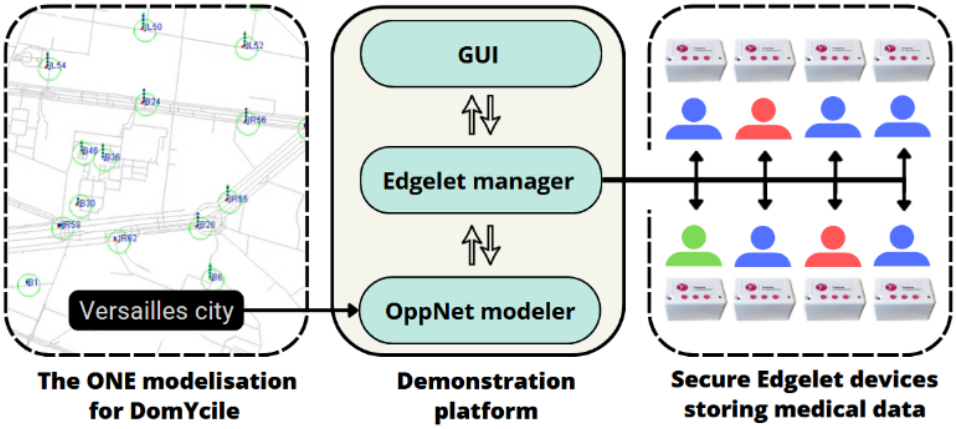

Figure 7.2: Architecture of the demonstration platform

[1] Concretely, we exported the map of the city of Versailles from OpenStreetMap and simulated the actual behaviors of the patients and health-workers in the DomYcile use case.

7.1.2 Realized Scenario

The demonstration allows to select a query among two representative ones of the DomYcile project:

- A Grouping Sets query [174] which allows multiple Group-By clauses to be evaluated within a single SQL query (to cross multiple statistics over a same cohort of patients).

- A K-means [168] followed by a Group By on the resulting clusters (to identify which characteristics most influence the dependency level of an elderly person).

Then, the platform suggests to improve the privacy of the QEP in order to reduce data exposure in case of TEEs compromise. To do so, following its intuition, the user is invited to adjust the horizontal and vertical partitioning parameters presented in Chapter 4. Regarding fault tolerance, the user can configure the probability of failure for each device as well as the timeout for messages in the OppNet, both of which affect the presumption of failure. This configuration adjustment allows the user to observe automatic changes in the QEP to maintain resiliency, providing insight into the impacts of the overcollection strategy (see Figure 7.3).

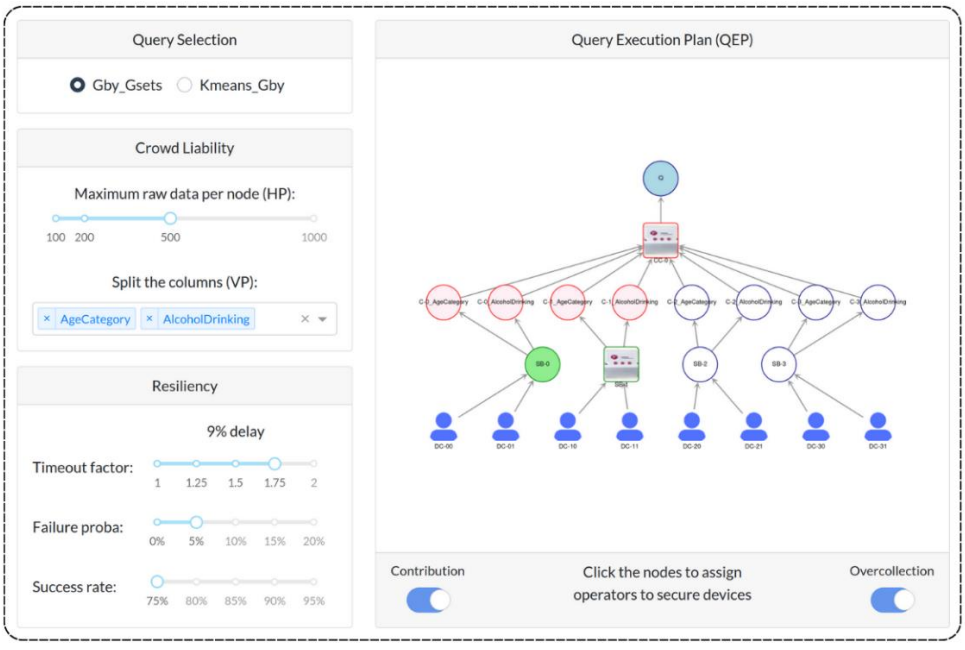

Figure 7.3: Configuration of a distributed QEP

Finally, the demonstration platform will proceed to the real-time execution of the resulting QEP on the boxes available for the demonstration (concrete edgelets), the rest of the operators being associated to a configurable number of simulated edgelets to attest scalability. The interactive GUI then provides a step-by-step visualization of the query execution, including input and output data for each operator as well as device failures and message losses.

7.1.3 Obtained Results

First, this demonstration illustrates a potential usage of the Edgelet computing paradigm through a real medical use case currently deployed in the field. It demonstrates that large-scale general-purpose computations can be performed over a set of opportunistic-connected devices while providing high security guarantees. Edgelet computing leverages the TEE security to perform computations on clear-text data (once decrypted locally), thus combining computation generality and scalability.

Second, the demonstration shows critical parts of the Edgelet computing internals related to (1) privacy preservation, thanks to the support of horizontal and vertical partitioning, and (2) resiliency, thanks to the Overcollection strategy, well adapted to sampling queries. Hence, we firmly believe that Edgelet computing opens up important opportunities in terms of personal data management.

7.2 Weakly Connected Personal Devices

Opportunistic Networks are an extreme case of uncertain communications in terms of latency and then fault presumption. However, uncertainty is inherent in any decentralized computation over a crowd of personal devices, because devices can be disconnected at will, be temporarily out of reach, or simply fail. In this second demonstration, we aim at enlarging the use cases targeted by Edgelet computing, considering that the solutions presented in this thesis apply whenever decentralized computations need to be performed among personal devices connected through "uncertain" communications.

The goal of this demonstration is twofold. First, it will exemplify the versatility of the approach by demonstrating the Edgelet computing mechanisms running on different weakly connected TEE-enabled devices (from PC with SGX up to smart objects with TPM). Second, it will present the internals of Edgelet computing and let the users play with important parameters related to resiliency and data privacy and observe the outcome by themselves.

7.2.1 Implemented Platform

The first objective of this demonstration is to present the computational mechanisms of the Edgelet framework using TEE-enabled devices ranging from high-end device (PC) to low-end device (home box). Therefore, we deploy the following hardware:

PC (Intel SGX). A laptop with an Intel Core i5-9400H 2.5GHz 4 Cores with SGX 1-FLC runs Ubuntu Linux 18.04 with SGX DCAP 1.14. The code is written on top of Open Enclave [175], an SDK for developing enclave applications in C/C++. Open Enclave provides support for Intel SGX as well as preview support for ARM TrustZone, thus aiming to generalize the development of enclave applications across TEEs.

Home boxes (TPM). As already presented in Section 7.1.1, these secure boxes incorporate notably a STM32F417 microcontroller dedicated to the management of personal data. This microcontroller is connected to a µ-SD card hosting the owner's raw data and to a tamper-proof Trusted Platform Module (TPM). This TPM secures the cryptographic keys and guarantees, during the secure boot, that the embedded code has not been tampered with.

The second objective is to present the internal aspects of the solution, we thus developed the following software:

- A graphical interface to interactively configure and visualize Edgelet queries.
- An Edgelet manager to orchestrate executions and communications between simulated and real edgelets (PC and home boxes).
- A web client accessible to users' smartphones via a QR code allowing them to monitor the processed data and interact in real-time with the execution.

7.2.2 Realized Scenario

In this demonstration, we will take the motivating example presented earlier. Let us assume that Santé Publique France (Recipient) wants to perform a set of queries on population health data to improve the quality of its services. Some individuals are equipped with a PC, others with a home box, but all are interconnected by uncertain communications. The scenario consists of two interactive parts related to the configuration and execution of Edgelet

computations. In the first part, the users will understand the impact of privacy and resiliency parameters on the QEPs. In the second part, they will follow the execution in real time and visualize the results obtained.

Part 1: QEP Configuration. The users are first invited to select one of two queries: either a Grouping Sets query [174] to cross multiple statistics over the same data sample, or a K-means [168] followed by a Group By to identify which characteristics most influence the dependency level of an elderly person. Then, following their intuition, the users can try to improve the privacy of the QEP of the selected query to reduce data exposure in case of TEEs compromise. To do so, they can adjust the horizontal and vertical partitioning parameters, by specifying the maximum number of raw data per edgelet and selecting the attribute pairs to be separated. Finally, the users can vary the failure probability value of the scenario and observe automatic changes in the execution plan to keep it resilient.

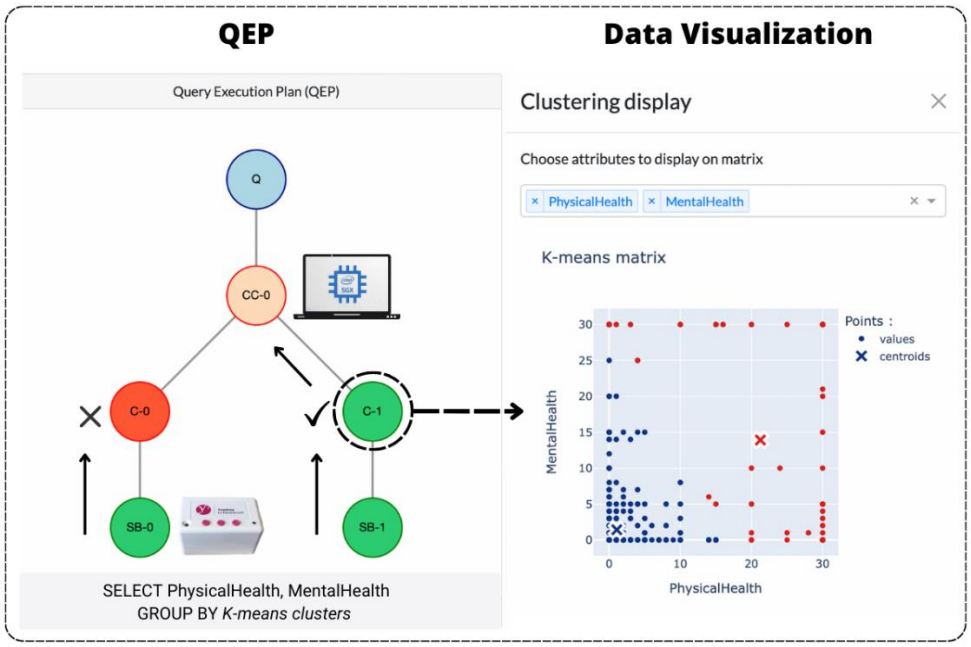

Figure 7.4: Data visualization for a distributed QEP

Part 2: Execution of an Edgelet computation. After the users have configured the query as desired, we proceed to its real-time execution on the heterogeneous personal devices available for the demonstration (PC or home boxes), the rest of the operators being associated to a configurable number of simulated edgelets to attest scalability. First, we launch the collection phase where the Snapshot Builders receive contributions from thousands of simulated Data Contributors and build representative snapshots. Next, the Edgelet platform redistributes the data and launches the computation phase

with the corresponding Computers. At each step, the user can interact with the execution using their own smartphone to analyze the input and output data. At the end, the aggregated data is transmitted to the Computing Combiners for the combination phase and the query is completed. In case of failures or disconnections, the users are able to directly identify the concerned edgelets on the QEP and understand the impacts on the execution (see Figure 7.4). For example, we can intentionally power off some concrete devices to generate a failure at will. In order to verify the results, the users can take the same dataset used with the distributed edgelets and run the processing centrally on the platform.

7.2.3 Obtained Results

The demonstration shows the internals of the Edgelet computing framework applied to the fully decentralized context and illustrates its usage through execution scenarios on multiple personal devices. It helps to answer the following questions:

Does Edgelet computing concretely make sense? The practical implementation on high-end and low-end personal devices demonstrates both its applicability and versatility. It shows that large-scale general-purpose computations can be performed over devices while providing high-security guarantees. This opens up important opportunities in terms of personal data management.

Can any form of computation be handled? Edgelet computing leverages the TEE security to perform computations on clear-text data (once decrypted locally). It can then combine computation generality – demonstrated by our demonstration queries – and scalability – demonstrated by the number of simulated edgelets –, contrary to homomorphic encryption, secure multiparty computation, or local differential privacy solutions (see Section 2.4 of the related works).

Is privacy protected whatever the attack? While highly difficult to implement, side-channel attacks on a TEE could compromise the confidentiality of the data manipulated on that TEE. Edgelet computing counter-measures are horizontal and vertical partitioning. Through the demo GUI, the users are able to visualize the distribution of data among the edgelets and measure the respective benefit of both types of partitioning. They also understand that only the results of the computations, i.e., the aggregated data, are transmitted (encrypted) to the successor operators.

Can a query always proceed despite the failures? Providing fault tolerance in a distributed context where messages are sent among weakly connected personal devices is a real challenge, either because they are down or because they are temporarily unavailable (e.g., individual's smartphone offline). The demonstration shows that the Overcollection strategy can answer this: the users can vary the failure context (e.g., disconnection probability) and see the impact on the overcollection degree as well as the effects on the results accuracy.

8 CONCLUSION

In the rapidly expanding landscape of digital technologies and data creation, it has become evident that the centralized models proposed by the web giants are reaching their limits. The need for privacy-preserving solutions is more crucial than ever, as privacy breaches and massive data collection continue to raise serious concerns. While data protection laws (e.g., GDPR) and the development of Personal Data Management Systems (PDMS) aim to empower individuals to regain control over their digital lives, it is difficult to establish a fully decentralized model that supports both personal and collective data usage. In this thesis, we explore the distributed use of these PDMS in an Opportunistic Network context, where messages are transferred from one device to another without the need for any infrastructure. The proposed approach enables the implementation of complex processing involving the data of thousands of individuals, while guaranteeing the security and fault tolerance of the executions.

8.1 SUMMARY OF THE CONTRIBUTIONS

As described in Chapter 3, the challenge of this research work is to combine the computational resources of personal devices (e.g., smartphones, personal computers, smart objects, etc.) to perform fully decentralized privacy-preserving queries within Opportunistic Networks. Our contributions are the following:

1. We define the Edgelet computing paradigm, leveraging the convergence between Trusted Execution Environments and Opportunistic Networks, as a new framework for performing complex processing on personal devices in a highly distributed, failure-prone, and infrastructure-less environment. We propose a shared responsibility model adapted to this framework, called Crowd Liability Model (CLM), to capture the liability shift from the data controller (in the GDPR sense) to the crowd. We then present the Query Execution Plans considered in this work and analyse the difficulties associated with their practical implementation. Based on this analysis, we formalize the problem to be solved with the definition of three properties of distributed systems, namely Confidentiality (security), Resiliency (liveness) and Validity (safety).

2. We propose a threat model dedicated to the CLM capturing the malicious attacks seeking to compromise the query executions. On this basis, we first design a protocol that establishes trust between the crowd members and the Recipient(s) of the queries. Second, we leverage the properties of TEEs to provide security mechanisms that preserve both the integrity of the decentralized executions and the confidentiality of the manipulated data.

3. In order to make the executions tolerant to devices failures and message losses, we propose three resiliency strategies: Backup, Overcollection and Hybrid. As these strategies have different impacts on the validity of results and confidentiality of the data, we provide a qualitative analysis of all of them including a taxonomy of query scenarios for which they are more or less suitable. The findings of this analysis serve as guidelines for determining the most appropriate execution model when designing a computation under the Edgelet computing paradigm.

4. We finally present quantitative evaluations of the proposed methods and strategies in order to validate them and study their limits. Furthermore, we propose two practical implementations of the Edgelet architecture that demonstrate both the genericity and security of the framework in real-world scenarios. While the first demonstration focuses on the medical use case of the DomYcile project, the second opens up new possibilities in terms of usage. Indeed, we show that the Edgelet paradigm can be applied to any environment composed of weakly connected personal devices, even heterogeneous ones.

8.2 PERSPECTIVES

To the best of our knowledge, this work is the first attempt to combine Personal Data Management Systems and Opportunistic Networks to propose fully decentralized privacy-preserving computations. Based on our proposed architecture, Edgelet computing, several research challenges remain to be investigated, including the following:

Optimization of resource consumption. A first challenge is to explore the multiple optimization tradeoffs that exist between data confidentiality, query success rate and (local and global) resource consumption. As we explain in Chapter 3, we chose an epidemic diffusion of message in the Opportunistic

Network. This simplistic routing protocol has the benefit of maximizing the message delivery rate and minimizing message latency, but at the cost of significant network congestion and overhead. A significant improvement would be to incorporate individuals' social patterns into routing (e.g., BUBBLE Rap [176]) and facilitate query processing by assigning operators to the most connected devices (e.g., doctors, teachers). Such optimizations would lead to revisit the solutions enforcing the CLM's Computation honesty, in particular the random assignment protocol to allow a (limited and controlled) degree of bias in favor of socially well-connected crowd members.

Management of long-lasting snapshots. A second challenge is to integrate long-lasting snapshots (i.e., persistent datasets) to support processes routinely used in data analysis. Such processes start with an initial set of exploration queries to capture data frequency distributions before running precise database queries, data mining or machine learning algorithms. Long-lasting snapshots could resort to specific indexing schemes to re-access sets of participating edgelets or materialized snapshot partitions kept (encrypted) on sets of edgelets. Obviously persistent data management would have various impacts on the properties of Confidentiality, Resiliency, and Validity. How to prevent data at rest from being the target of malicious attacks? How to preserve the consistency of successive queries in a fully decentralized environment?

Improvement of the validity of iterative algorithms. A third challenge is to study classes of iterative algorithms compatible with Edgelet computing for which it is possible to prove the strict validity of the results. For example, as mentioned in Chapter 5, we observe that the Apriori algorithm executed in Edgelet can produce exactly the same results as a centralized execution. To go further and deepen our first experimental evaluations, it would be essential to theoretically prove the convergence of the algorithms. The research work of Lian et al. [169] on the convergence of the Asynchronous Decentralized Parallel Stochastic Gradient Descent particularly illustrates the interest of this type of proof. Thus, in the same direction, a theoretical and systemic analysis of iterative processing in Edgelet would be an important asset for the deployment of new applications in this context.

Deployment of an operational platform. A fourth challenge is to deploy the Edgelet computing framework in real world. As detailed in Chapter 7, we have demonstrated that our research work is applicable to the DomYcile project, with the potential to bring real added value in terms of usage. The next step would be to valorize this work with an effective implementation of the Edgelet mechanisms within the deployed boxes. We will then have to deal

with the difficulties associated with the realities of the field, including the heterogeneity of the boxes' hardware. Indeed, the version currently deployed has no Internet connection while the new version will be equipped with a 4G chip. The environment will therefore be halfway between our two demonstrations, composed of both OppNet communicating devices and weakly connected ones.

BIBLIOGRAPHY

[1] D. Reinsel, J. Gantz, J. Rydning, and others, "The digitization of the world from edge to core," *IDC white paper*, vol. 13, 2018.

[2] "Cisco Annual Internet Report (2018–2023) White Paper," Mar. 2020. https://tinyurl.com/cisco-internet-report

[3] D. D. Hirsch, "The glass house effect: Big Data, the new oil, and the power of analogy," *Me. L. Rev.*, vol. 66, p. 373, 2013.

[4] "Firefox Monitor." https://monitor.firefox.com

[5] L. Poitras, "Citizenfour," *Lectures, publications reçues*, 2015.

[6] G. Venkatadri *et al.*, "Privacy Risks with Facebook's PII-Based Targeting: Auditing a Data Broker's Advertising Interface," in *2018 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA: IEEE, May 2018, pp. 89–107.

[7] M. Silva, L. S. de Oliveira, A. Andreou, P. O. V. de Melo, O. Goga, and F. Benevenuto, "Facebook Ads Monitor: An Independent Auditing System for Political Ads on Facebook." arXiv, Jan. 31, 2020.

[8] "Regulation EU 2016/679 of the European Parliament and of the Council." http://data.europa.eu/eli/reg/2016/679

[9] "Blue Button," *Wikipedia*. https://en.wikipedia.org/wiki/Blue_Button

[10] "MesInfos." https://fing.org/toutes-les-actions/mesinfos.html

[11] "Google Drive." https://drive.google.com

[12] "Signal Messenger: Speak Freely." https://signal.org

[13] "StartMail - Private email you can trust." https://www.startmail.com

[14] "Tresorit." https://tresorit.com

[15] N. Anciaux *et al.*, "Personal Data Management Systems: The security and functionality standpoint," *Information Systems*, vol. 80, pp. 13–35, 2019.

[16] "Cozy Cloud." https://cozy.io

[17] R. Ladjel, N. Anciaux, P. Pucheral, and G. Scerri, "Trustworthy Distributed Computations on Personal Data Using Trusted Execution Environments," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, Rotorua, New Zealand: IEEE, Aug. 2019, pp. 381–388.

[18] J. Mirval, L. Bouganim, and I. S. Popa, "Practical Fully-Decentralized Secure Aggregation for Personal Data Management Systems," in *SSDBM 2021: 33rd International Conference on Scientific and Statistical Database Management, Tampa, FL, USA, July 6-7, 2021*, ACM, 2021, pp. 259–264.

[19] L. Bouganim, J. Loudet, and I. Sandu Popa, "Highly distributed and privacy-preserving queries on personal data management systems," *The VLDB Journal*, pp. 1–31, 2022.

[20] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking: data forwarding in disconnected mobile ad hoc networks," *IEEE Commun. Mag.*, vol. 44, no. 11, pp. 134–141, Nov. 2006.

[21] "DomYcile Project: https://tinyurl.com/domycile, Salon E-Tonomy: https://tinyurl.com/e-tonomy," 2020.

[22] "HIPPOCAD." https://www.hippocad.com

[23] "Proposal for a Proposal for a Regulation of the European Parliament and of the Council on European data governance (Data Governance Act)," 2020. https://tinyurl.com/data-governance-act

[24] M. Conti, S. Giordano, M. May, and A. Passarella, "From opportunistic networks to opportunistic computing," *IEEE Commun. Mag.*, vol. 48, no. 9, pp. 126–139, Sep. 2010.

[25] L. Javet, N. Anciaux, L. Bouganim, and P. Pucheral, "Edgelet Computing: Pushing Query Processing and Liability at the Extreme Edge of the Network," in *22nd IEEE International Symposium on Cluster, Cloud and Internet Computing, CCGrid 2022, Taormina, Italy, May 16-19, 2022*, IEEE, 2022, pp. 160–169.

[26] L. Javet, N. Anciaux, L. Bouganim, L. Lamoureux, and P. Pucheral, "Secure Computations in Opportunistic Networks: An Edgelet Demonstration with a Medical Use-Case," in *PerCom 2023-21st IEEE International Conference on Pervasive Computing and Communications*, 2023.

[27] L. Javet, N. Anciaux, L. Bouganim, L. Lamoureux, and P. Pucheral, "Pushing Edge Computing one Step Further: Resilient and Privacy-Preserving Processing on Personal Devices," in *Proceedings 26th International Conference on Extending Database Technology, EDBT 2023, Ioannina, Greece, March 28-31, 2023*, OpenProceedings.org, 2023, pp. 835–838.

[28] "PlugDB." https://project.inria.fr/plugdb/en/

[29] "Dropbox." https://www.dropbox.com

[30] "OneDrive." https://onedrive.com

[31] "SpiderOak." https://spideroak.com

[32] "Sync.com." https://www.sync.com

[33] "Digi.me." https://digi.me

[34] "Solid." https://solidproject.org

[35] Y.-A. de Montjoye, E. Shmueli, S. S. Wang, and A. S. Pentland, "openPDS: Protecting the Privacy of Metadata through SafeAnswers," *PLoS ONE*, vol. 9, no. 7, p. e98790, Jul. 2014.

[36] A. Chaudhry *et al.*, "Personal data: thinking inside the box," in *Proceedings of The Fifth Decennial Aarhus Conference on Critical Alternatives, 2015, Aarhus, Denmark, August 17-21, 2015*, Aarhus University Press / ACM, 2015, pp. 29–32.

[37] "Personium." https://personium.io

[38] "Nextcloud." https://nextcloud.com

[39] "Amber X." https://www.myamberlife.com

[40] "Helixee." https://www.helixee.me/home/

[41] "Meet Lima." https://meetlima.com

[42] T. Allard *et al.*, "Secure personal data servers: a vision paper," *Proc. VLDB Endow.*, vol. 3, no. 1–2, pp. 25–35, Sep. 2010.

[43] N. Anciaux, P. Bonnet, L. Bouganim, B. Nguyen, I. S. Popa, and P. Pucheral, "Trusted Cells: A Sea Change for Personal Data Services," in *Sixth Biennial Conference on Innovative Data Systems Research, CIDR 2013, Asilomar, CA, USA, January 6-9, 2013, Online Proceedings*, 2013.

[44] N. Anciaux, M. Benzine, L. Bouganim, K. Jacquemin, P. Pucheral, and S. Yin, "Restoring the Patient Control over Her Medical History," in *2008 21st IEEE International Symposium on Computer-Based Medical Systems*, Jun. 2008, pp. 132–137.

[45] N. Anciaux *et al.*, "A Tamper-Resistant and Portable Healthcare Folder," *International Journal of Telemedicine and Applications*, vol. 2008, pp. 1–9, 2008.

[46] N. Anciaux *et al.*, "Managing Personal Health Records in an Infrastructure-Weak Environment," in *e-Infrastructure and e-Services*, Cham: Springer International Publishing, 2016, pp. 178–191.

[47] S. Lallali, N. Anciaux, I. Sandu Popa, and P. Pucheral, "Supporting secure keyword search in the personal cloud," *Information Systems*, vol. 72, pp. 1–26, Dec. 2017.

[48] N. Anciaux, B. Nguyen, and I. S. Popa, "Personal Data Management with Secure Hardware: How to Keep Your Data at Hand," in *2013 IEEE 14th International Conference on Mobile Data Management*, Milan, Italy: IEEE, Jun. 2013, pp. 1–2.

[49] N. Anciaux, B. Nguyen, and I. S. Popa, "Tutorial: Managing Personal Data with Strong Privacy Guarantees," in *Proceedings of the 17th International Conference on Extending Database Technology, EDBT 2014, Athens, Greece, March 24-28, 2014*, OpenProceedings.org, 2014, pp. 672–673.

[50] N. Anciaux, L. Bouganim, P. Pucheral, Y. Guo, L. Le Folgoc, and S. Yin, "MILo-DB: a personal, secure and portable database machine," *Distrib Parallel Databases*, vol. 32, no. 1, pp. 37–63, Mar. 2014.

[51] D. H. T. That, I. S. Popa, K. Zeitouni, and C. Borcea, "PAMPAS: Privacy-Aware Mobile Participatory Sensing Using Secure Probes," in *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*, Budapest Hungary: ACM, Jul. 2016, pp. 1–12.

[52] Q.-C. To, B. Nguyen, and P. Pucheral, "Private and Scalable Execution of SQL Aggregates on a Secure Decentralized Architecture," *ACM Trans. Database Syst.*, vol. 41, no. 3, pp. 1–43, Aug. 2016.

[53] R. Ladjel, N. Anciaux, P. Pucheral, and G. Scerri, "A Manifest-Based Framework for Organizing the Management of Personal Data at the Edge of the Network," in *Information Systems Development: Information Systems Beyond 2020, ISD 2019 Proceedings, Toulon, France,* 2019.

[54] R. Carpentier, F. Thiant, I. S. Popa, N. Anciaux, and L. Bouganim, "An Extensive and Secure Personal Data Management System Using SGX," in *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK,* 2022, p. 2:570-2:573.

[55] R. Carpentier, I. Sandu Popa, and N. Anciaux, "Data Leakage Mitigation of User-Defined Functions on Secure Personal Data Management Systems," in *34th International Conference on Scientific and Statistical Database Management*, Copenhagen Denmark: ACM, Jul. 2022, pp. 1–12.

[56] M. Conti and S. Giordano, "Mobile ad hoc networking: milestones, challenges, and new research directions," *IEEE Commun. Mag.*, vol. 52, no. 1, pp. 85–96, Jan. 2014.

[57] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, in SIGCOMM '03. New York, NY, USA: Association for Computing Machinery, 2003, pp. 27–34.

[58] C. Boldrini, K. Lee, M. Önen, J. Ott, and E. Pagani, "Opportunistic networks," *Computer Communications*, vol. 48, pp. 1–4, Jul. 2014.

[59] M. Alajeely, R. Doss, and A. Ahmad, "Routing Protocols in Opportunistic Networks – A Survey," *IETE Technical Review*, vol. 35, no. 4, pp. 369–387, Jul. 2018.

[60] Y. Cao and Z. Sun, "Routing in Delay/Disruption Tolerant Networks: A Taxonomy, Survey and Challenges," *IEEE Commun. Surv. Tutorials*, vol. 15, no. 2, pp. 654–677, 2013.

[61] A. Vahdat and D. Becker, "Epidemic Routing for Partially-Connected Ad Hoc Networks," 2000.

[62] S. Jain, K. R. Fall, and R. K. Patra, "Routing in a delay tolerant network," in *Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 30 - September 3, 2004, Portland, Oregon, USA*, ACM, 2004, pp. 145–158.

[63] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic Routing in Intermittently Connected Networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, Jul. 2003.

[64] S. Trifunovic, S. T. Kouyoumdjieva, B. Distl, L. Pajevic, G. Karlsson, and B. Plattner, "A Decade of Research in Opportunistic Networks: Challenges, Relevance, and Future Directions," *IEEE Commun. Mag.*, vol. 55, no. 1, pp. 168–173, 2017.

[65] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet," in *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X), San Jose, California, USA, October 5-9, 2002*, ACM Press, 2002, pp. 96–107.

[66] V. G. Menon and P. M. J. Prathap, "Comparative analysis of opportunistic routing protocols for underwater acoustic sensor networks," in *2016 International Conference on Emerging Technological Trends (ICETT)*, Kollam, India: IEEE, Oct. 2016, pp. 1–5.

[67] B. Han, P. Hui, V. S. A. Kumar, M. V. Marathe, J. Shao, and A. Srinivasan, "Mobile Data Offloading through Opportunistic Communications and Social Participation," *IEEE Trans. on Mobile Comput.*, vol. 11, no. 5, pp. 821–834, May 2012.

[68] "Opengarden's firechat app." https://fr.wikipedia.org/wiki/FireChat

[69] M. Saloni, C. Julien, A. L. Murphy, and G. P. Picco, "Lasso: A device-to-device group monitoring service for smart cities," in *2017 International Smart Cities Conference (ISC2)*, Wuxi, China: IEEE, Sep. 2017, pp. 1–6.

[70] N. Vastardis and Kun Yang, "Mobile Social Networks: Architectures, Social Properties, and Key Research Challenges," *IEEE Commun. Surv. Tutorials*, vol. 15, no. 3, pp. 1355–1371, 2013.

[71] H. Li, K. Ota, M. Dong, and M. Guo, "Mobile Crowdsensing in Software Defined Opportunistic Networks," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 140–145, Jun. 2017.

[72] A. Lindgren and P. Hui, "The Quest for a Killer App for Opportunistic and Delay Tolerant Networks: (Invited Paper)," in *Proceedings of the 4th ACM Workshop on Challenged Networks*, in CHANTS '09. New York, NY, USA: Association for Computing Machinery, 2009, pp. 59–66.

[73] "Uepaa! - alpine safety app." https://safety.uepaa.ch

[74] "Starlink." https://www.starlink.com

[75] F. Guidec, Y. Maheo, P. Launay, L. Touseau, and C. Nous, "Bringing Opportunistic Networking to Smartphones: a Pragmatic Approach," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, Madrid, Spain: IEEE, Jul. 2021, pp. 574–579.

[76] J. Dede *et al.*, "Simulating Opportunistic Networks: Survey and Future Directions," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 2, pp. 1547–1573, 2017.

[77] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA: ICST, 2009.

[78] "ns-3." https://www.nsnam.org

[79] M. Conti, A. Passarella, and S. K. Das, "The Internet of People (IoP): A new wave in pervasive mobile computing," *Pervasive Mob. Comput.*, vol. 41, pp. 1–27, 2017.

[80] M. J. Fischer, N. A. Lynch, and M. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *J. ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[81] A. Benchi, P. Launay, and F. Guidec, "Solving Consensus in Opportunistic Networks," in *Proceedings of the 2015 International Conference on Distributed Computing and Networking, ICDCN 2015, Goa, India, January 4-7, 2015*, ACM, 2015, pp. 1–10.

[82] R. Dragan, R.-I. Ciobanu, and C. Dobre, "Leader Election in Opportunistic Networks," in *2017 16th International Symposium on Parallel and Distributed Computing (ISPDC)*, Innsbruck: IEEE, Jul. 2017, pp. 157–164.

[83] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.

[84] I. C. L. Ng and S. Y. L. Wakenshaw, "The Internet-of-Things: Review and research directions," *International Journal of Research in Marketing*, vol. 34, no. 1, pp. 3–21, 2017.

[85] "Industrial Automation." https://tinyurl.com/industrial-automation

[86] D. O'Keeffe, T. Salonidis, and P. R. Pietzuch, "Frontier: Resilient Edge Processing for the Internet of Things," *Proc. VLDB Endow.*, vol. 11, no. 10, pp. 1178–1191, 2018.

[87] S. Zeuch *et al.*, "The NebulaStream Platform for Data and Application Management in the Internet of Things," in *10th Conference on Innovative Data Systems Research, CIDR 2020, Amsterdam, The Netherlands,* 2020.

[88] L. González-Manzano, J. M. de Fuentes, S. Pastrana, P. Peris-Lopez, and L. Hernández-Encinas, "PAgIoT – Privacy-preserving Aggregation protocol for Internet of Things," *Journal of Network and Computer Applications*, vol. 71, pp. 59–71, Aug. 2016.

[89] K. Nellore and G. Hancke, "A Survey on Urban Traffic Management System Using Wireless Sensor Networks," *Sensors*, vol. 16, no. 2, p. 157, Jan. 2016.

[90] H. Alwan and A. Agarwal, "A Survey on Fault Tolerant Routing Techniques in Wireless Sensor Networks," in *2009 Third International Conference on Sensor Technologies and Applications*, Athens, Greece, 2009, pp. 366–371.

[91] C. Berger, P. Eichhammer, H. P. Reiser, J. Domaschka, F. J. Hauck, and G. Habiger, "A Survey on Resilience in the IoT: Taxonomy, Classification, and Discussion of Resilience Mechanisms," *ACM Comput. Surv.*, vol. 54, no. 7, pp. 1–39, Sep. 2022.

[92] D. G. Murray, E. Yoneki, J. Crowcroft, and S. Hand, "The case for crowd computing," in *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds*, New Delhi India: ACM, Aug. 2010, pp. 39–44.

[93] K. Parshotam, "Crowd computing: a literature review and definition," in *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, in SAICSIT '13. New York, NY, USA: Association for Computing Machinery, Oct. 2013, pp. 121–130.

[94] B. Guo *et al.*, "Mobile Crowd Sensing and Computing: The Review of an Emerging Human-Powered Sensing Paradigm," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 1–31, Sep. 2015.

[95] E. Estellés-Arolas and F. González-Ladrón-de-Guevara, "Towards an integrated crowdsourcing definition," *Journal of Information Science*, vol. 38, no. 2, pp. 189–200, 2012.

[96] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[97] D. Christin, "Privacy in mobile participatory sensing: Current trends and future challenges," *Journal of Systems and Software*, vol. 116, pp. 57–68, Jun. 2016.

[98] L. Pournajaf, D. A. Garcia-Ulloa, L. Xiong, and V. Sunderam, "Participant Privacy in Mobile Crowd Sensing Task Management: A Survey of Methods and Challenges," *SIGMOD Rec.*, vol. 44, no. 4, pp. 23–34, May 2016.

[99] Z. Wang *et al.*, "Personalized Privacy-Preserving Task Allocation for Mobile Crowdsensing," *IEEE Trans. on Mobile Comput.*, vol. 18, no. 6, pp. 1330–1341, Jun. 2019.

[100] M. Brahem, G. Scerri, N. Anciaux, and V. Issarny, "Consent-driven Data Reuse in Multi-tasking Crowdsensing Systems: A Privacy-by-Design Solution," *Pervasive and Mobile Computing*, vol. 83, p. 101614, Jul. 2022.

[101] L. Ponciano and F. Brasileiro, "Agreement-based credibility assessment and task replication in human computation systems," *Future Generation Computer Systems*, vol. 87, pp. 159–170, Oct. 2018.

[102] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, Istanbul Turkey: ACM, Aug. 2012, pp. 173–184.

[103] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. S. Shen, "Providing Task Allocation and Secure Deduplication for Mobile Crowdsensing via Fog Computing," *IEEE Trans. Dependable and Secure Comput.*, vol. 17, no. 3, pp. 581–594, 2020.

[104] A. J. Ferrer, J. M. Marquès, and J. Jorba, "Towards the Decentralised Cloud: Survey on Approaches and Challenges for Mobile, Ad hoc, and Edge Computing," *ACM Comput. Surv.*, vol. 51, no. 6, p. 111:1-111:36, 2019.

[105] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[106] M. Yannuzzi, R. Milito, R. Serral-Gracia, D. Montero, and M. Nemirovsky, "Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing," in *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Athens, Greece: IEEE, Dec. 2014, pp. 325–329.

[107] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge Computing for Autonomous Driving: Opportunities and Challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, Aug. 2019.

[108] A. S. Albahri *et al.*, "IoT-based telemedicine for disease prevention and health promotion: State-of-the-Art," *Journal of Network and Computer Applications*, vol. 173, p. 102873, Jan. 2021.

[109] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009.

[110] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data Security and Privacy-Preserving in Edge Computing Paradigm: Survey and Open Issues," *IEEE Access*, vol. 6, pp. 18209–18237, 2018.

[111] J. Zhao, R. Mortier, J. Crowcroft, and L. Wang, "Privacy-Preserving Machine Learning Based Data Analytics on Edge Devices," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, New Orleans LA USA: ACM, Dec. 2018, pp. 341–346.

[112] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "The Extended Cloud: Review and Analysis of Mobile Edge Computing and Fog From a Security and Resilience Perspective," *IEEE J. Select. Areas Commun.*, vol. 35, no. 11, pp. 2586–2595, Nov. 2017.

[113] R. Roman, J. Lopez, and M. Mambo, "Mobile Edge Computing, Fog et al.: A Survey and Analysis of Security Threats and Challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, Jan. 2018.

[114] J. Grover and R. M. Garimella, "Reliable and Fault-Tolerant IoT-Edge Architecture," in *2018 IEEE SENSORS*, New Delhi: IEEE, Oct. 2018, pp. 1–4.

[115] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Comput. Surv.*, vol. 36, no. 4, pp. 335–371, Dec. 2004.

[116] "BOINC." https://boinc.berkeley.edu

[117] "World Community Grid." https://www.worldcommunitygrid.org

[118] M. Ghaffari, N. Ghadiri, M. H. Manshaei, and M. S. Lahijani, "P4QS: A Peer-to-Peer Privacy Preserving Query Service for Location-Based Mobile Applications," *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 9458–9469, Oct. 2017.

[119] D. Stutzbach and R. Rejaie, "Understanding churn in peer-to-peer networks," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, Rio de Janeriro Brazil: ACM, 2006, pp. 189–202.

[120] E. Spaho, A. Barolli, F. Xhafa, and L. Barolli, "P2P Data Replication: Techniques and Applications," in *Modeling and Processing for Next-Generation Big-Data Technologies*, Cham: Springer International Publishing, 2015, pp. 145–166.

[121] M. Kaddoura, N. Bahr, and E. Gambucci, "SH-P2P: Self-Healing Peer-to-Peer Network with Optimal Multicast Routing," in *2022 IEEE International Conference on Electro Information Technology (eIT)*, Mankato, MN, USA: IEEE, May 2022, pp. 027–031.

[122] P. Kairouz *et al.*, "Advances and Open Problems in Federated Learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.

[123] "Machine Learning Ledger Orchestration for Drug Discovery," *CORDIS | European Commission*, 2019. https://cordis.europa.eu/project/id/831472

[124] A. Hard *et al.*, "Federated Learning for Mobile Keyboard Prediction." arXiv, Feb. 28, 2019.

[125] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019, pp. 739–753.

[126] E. Cyffers, M. Even, A. Bellet, and L. Massoulié, "Muffliato: Peer-to-Peer Privacy Amplification for Decentralized Optimization and Averaging." arXiv, Oct. 19, 2022.

[127] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, Mar. 2021.

[128] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated Multi-Task Learning," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 4424–4434.

[129] R. L. Rivest, L. Adleman, M. L. Dertouzos, and others, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[130] R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[131] C. Gentry, "A fully homomorphic encryption scheme," PhD Thesis, Stanford University, USA, 2009.

[132] P. Martins, L. Sousa, and A. Mariano, "A Survey on Fully Homomorphic Encryption: An Engineering Perspective," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 1–33, Nov. 2018.

[133] A. C. Yao, "Protocols for secure computations," in *23rd annual symposium on foundations of computer science (sfcs 1982)*, IEEE, 1982, pp. 160–164.

[134] E. Saleh, A. Alsa'deh, A. Kayed, and C. Meinel, "Processing Over Encrypted Data: Between Theory and Practice," *SIGMOD Rec.*, vol. 45, no. 3, pp. 5–16, Dec. 2016.

[135] A. C.-C. Yao, "How to Generate and Exchange Secrets (Extended Abstract)," in *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, IEEE Computer Society, 1986, pp. 162–167.

[136] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)," in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, J. Simon, Ed., ACM, 1988, pp. 1–10.

[137] A. Shamir, "How to Share a Secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[138] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty Computation from Somewhat Homomorphic Encryption," in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 643–662.

[139] T. Allard, G. Hébrail, F. Masseglia, and E. Pacitti, "Chiaroscuro: Transparency and Privacy for Massive Personal Time-Series Clustering," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, Melbourne Victoria Australia: ACM, May 2015, pp. 779–794.

[140] J. Bater, G. Elliott, C. Eggen, S. Goel, A. N. Kho, and J. Rogers, "SMCQL: Secure Query Processing for Private Data Networks," *Proc. VLDB Endow.*, vol. 10, no. 6, pp. 673–684, 2017.

[141] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. D. Smith, "What Can We Learn Privately?," *SIAM J. Comput.*, vol. 40, no. 3, pp. 793–826, 2011.

[142] C. Dwork, "Differential Privacy," in *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds., in Lecture Notes in Computer Science, vol. 4052. Springer, 2006, pp. 1–12.

[143] P. Samarati and L. Sweeney, "Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression," 1998.

[144] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "L-diversity: privacy beyond k-anonymity," in *22nd International Conference on Data Engineering (ICDE'06)*, Atlanta, GA, USA: IEEE, 2006, pp. 24–24.

[145] N. Li, T. Li, and S. Venkatasubramanian, "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity," in *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, R. Chirkova, A. Dogac, M. T. Özsu, and T. K. Sellis, Eds., IEEE Computer Society, 2007, pp. 106–115.

[146] T. T. Nguyên, X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin, "Collecting and Analyzing Data from Smart Device Users with Local Differential Privacy." arXiv, Jun. 16, 2016.

[147] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted Execution Environment: What It is, and What It is Not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, IEEE, Aug. 2015, pp. 57–64.

[148] S. Pinto and N. Santos, "Demystifying Arm TrustZone: A Comprehensive Survey," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–36, Feb. 2019.

[149] L. Guan *et al.*, "TrustShadow: Secure Execution of Unmodified Applications with ARM TrustZone," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, Niagara Falls New York USA: ACM, Jun. 2017, pp. 488–501.

[150] "Android Keystore." https://tinyurl.com/android-keystore

[151] S. Wan, M. Sun, K. Sun, N. Zhang, and X. He, "RusTEE: Developing Memory-Safe ARM TrustZone Applications," in *Annual Computer Security Applications Conference*, Austin USA: ACM, Dec. 2020, pp. 442–453.

[152] I. Anati, S. Gueron, S. Johnson, and V. Scarlata, "Innovative technology for CPU based attestation and sealing," in *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*, ACM New York, NY, USA, 2013.

[153] V. Costan and S. Devadas, "Intel SGX Explained," *IACR Cryptol. ePrint Arch.*, p. 86, 2016.

[154] "Trusted Platform Module." https://trustedcomputinggroup.org/resource/tpm-library-specification/

[155] H. Raj *et al.*, "fTPM: A Software-Only Implementation of a TPM Chip," in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, USENIX Association, 2016, pp. 841–856.

[156] W. Wang *et al.*, "Leaky Cauldron on the Dark Land: Understanding Memory Side-Channel Hazards in SGX," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds., ACM, 2017, pp. 2421–2434.

[157] J. V. Bulck *et al.*, "Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, W. Enck and A. P. Felt, Eds., USENIX Association, 2018, pp. 991–1008.

[158] S. van Schaik, A. Kwong, D. Genkin, and Y. Yarom, "SGAxe: How SGX Fails in Practice," 2020.

[159] F. Tramer, F. Zhang, H. Lin, J.-P. Hubaux, A. Juels, and E. Shi, "Sealed-Glass Proofs: Using Transparent Enclaves to Prove and Sell Knowledge," in *EuroS&P*, IEEE, 2017, pp. 19–34.

[160] "Shared Responsibility Model - Amazon Web Services (AWS)." https://aws.amazon.com/compliance/shared-responsibility-model/

[161] "Closed-world assumption," *Wikipedia*. https://en.wikipedia.org/wiki/Closed-world_assumption

[162] M. Herr, N. Sirven, H. Grondin, S. Pichetti, and C. Sermet, "Frailty, polypharmacy, and potentially inappropriate medications in old people: findings in a representative sample of the French population," *Eur J Clin Pharmacol*, vol. 73, no. 9, pp. 1165–1172, 2017.

[163] "The Survey of Health, Ageing and Retirement in Europe (SHARE)." http://www.share-project.org/

[164] J. Ménétrey, C. Göttel, M. Pasin, P. Felber, and V. Schiavoni, "An Exploratory Study of Attestation Mechanisms for Trusted Execution Environments." arXiv, Apr. 15, 2022.

[165] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso, "Understanding replication in databases and distributed systems," in *Proceedings 20th IEEE International Conference on Distributed Computing Systems*, Taipei, Taiwan: IEEE Comput. Soc, 2000, pp. 464–474.

[166] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *19th International Conference on Computational Statistics, COMPSTAT 2010, Paris, France, August 22-27, 2010 - Keynote, Invited and Contributed Papers*, Y. Lechevallier and G. Saporta, Eds., Physica-Verlag, 2010, pp. 177–186.

[167] A. Savasere, E. Omiecinski, and S. B. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases," in *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland*, U. Dayal, P. M. D. Gray, and S. Nishio, Eds., Morgan Kaufmann, 1995, pp. 432–444.

[168] I. S. Dhillon and D. S. Modha, "A Data-Clustering Algorithm on Distributed Memory Multiprocessors," in *Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD, August 15, 1999, San Diego, CA, USA, revised papers*, M. J. Zaki and C.-T. Ho, Eds., in Lecture Notes in Computer Science, vol. 1759. Springer, 1999, pp. 245–260.

[169] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous Decentralized Parallel Stochastic Gradient Descent," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, J. G. Dy and A. Krause, Eds., in Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 3049–3058.

[170] T. Brijs, "Retail market basket data set," in *Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, 2003.

[171] "OpenML - Adult Income." http://www.openml.org/d/1590

[172] "Scikit-learn - California Housing." https://tinyurl.com/california-housing-dataset

[173] "Dash Python." https://dash.plotly.com

[174] "Group By Grouping Sets, Snowflake Documentation," 2023. https://tinyurl.com/grouping-sets

[175] "Open Enclave SDK," 2023. https://openenclave.io/sdk/

[176] P. Hui, J. Crowcroft, and E. Yoneki, "BUBBLE Rap: Social-Based Forwarding in Delay-Tolerant Networks," *IEEE Trans. Mob. Comput.*, vol. 10, no. 11, pp. 1576–1589, 2011.