Intermodal data transport for massive offloading of conventional data networks

Benjamin Baron

UPMC
SORBONNE UNIVERSITÉS

Thèse de doctorat

Transport intermodal de données massives pour le délestage des réseaux d'infrastructure

Benjamin **Baron**

pour obtenir le grade de
Docteur de l'Université Pierre et Marie Curie

École doctorale
INFORMATIQUE, TÉLÉCOMMUNICATIONS ET ÉLECTRONIQUE

à soutenir le 11 Octobre 2016 devant le jury composé de :

Rapporteurs :	Marco Fiore	Chercheur, CNR–IEIIT
	Sidi-Mohammed Senouci	Professeur, Université de Bourgogne
Examinateurs :	André-Luc Beylot	Professeur, ENSEEIHT
	Serge Fdida	Professeur, UPMC Sorbonne Universités
	Olivier Festor	Professeur, TELECOM Nancy
	Hubert Tardieu	CEO Advisor, Atos
Encadrants :	Prométhée Spathis	Maître de Conférence, UPMC Sorbonne Universités
	Marcelo Dias de Amorim	Directeur de recherche, CNRS/LIP6

This page intentionally left blank

UPMC
SORBONNE UNIVERSITÉS

PhD Thesis

Intermodal data transport for massive offloading of conventional data networks

Benjamin **Baron**

Submitted for the degree of
Doctor of Science of the Université Pierre et Marie Curie

Doctoral school
INFORMATIQUE, TÉLÉCOMMUNICATIONS ET ÉLECTRONIQUE

Thesis to be defended on October 11th, 2016

Reviewers:	Marco Fiore	Researcher, CNR–IEIIT
	Sidi-Mohammed Senouci	Professor, Université de Bourgogne
Examiners:	André-Luc Beylot	Professor, ENSEEIHT
	Serge Fdida	Professor, UPMC Sorbonne Universités
	Olivier Festor	Professor, TELECOM Nancy
	Hubert Tardieu	CEO Advisor, Atos
Advisors:	Prométhée Spathis	Associate Professor, UPMC Sorbonne Universités
	Marcelo Dias de Amorim	Research Director, CNRS/LIP6

Remerciements

Cette thèse aura été une belle aventure pour moi, elle m'a permis de m'enrichir d'un point de vue personnel et scientifique, en m'apportant de nouveaux regards sur le monde qui m'entoure. Même si son parcours a été parsemé d'obstacles, j'ai appris qu'il est toujours possible d'aller plus loin et de trouver un (ou même plusieurs) moyen(s) pour contourner les difficultés. Une thèse ne se fait pas tout seul, c'est pourquoi je tiens à remercier beaucoup de personnes qui m'ont soutenu jusqu'à son terme (j'espère ne pas en oublier).

Une thèse ne serait rien sans encadrants et sur ce point, j'ai eu beaucoup de chance d'avoir été encadré par Marcelo Dias de Amorim et Prométhée Spathis. C'est vrai que c'est un travail sur le long terme qui peut par de nombreux aspects s'apparenter à un rallye, avec ses routes tortueuses, ses dénivelés et épingles à cheveux. Mais avec un bon 4x4 (enfin 3x3 ici), on arrive à tout faire. Merci pour m'avoir guidé tout au long de cette belle expérience qui m'a permis de mûrir scientifiquement, mais aussi humainement.

Je voulais aussi remercier Sidi-Mohammed Senouci et Marco Fiore qui ont bien voulu commenter ma thèse. Je suis conscient qu'ils ont passé beaucoup de temps à la lire et à rédiger leur rapport. Leurs remarques m'ont permis d'améliorer la qualité du manuscrit et de la présentation. Je tenais aussi à remercier Hubert Tardieu, André-Luc Beylot, Olivier Festor et Serge Fdida pour avoir accepté de faire partie de mon jury et évaluer les travaux que je propose.

Je voulais aussi remercier toutes les personnes avec qui j'ai eu la chance de collaborer. J'ai entamé mes premières collaborations avec Hervé Rivano dès mon stage de Master, avec qui j'ai pu apprendre beaucoup sur les techniques d'optimisation. Je voulais aussi remercier Luis Costa et Miguel Campista pour m'avoir permis de participer à leurs recherches sur la virtualisation. I had the pleasure of meeting Yannis Viniotis during his summer visit in Paris. Thank you for all of the discussions we had and the time that you spent helping me improving my work. I was also very fortunate to have the opportunity to visit Mostafa Ammar in the US. It was truly a memorable experience in which I learned a great deal about the American style of research in the beautiful state of Georgia. Thank you for this opportunity and all the insightful discussions we had. I also wanted to thank all my colleagues in GeorgiaTech for the many discussions we've had and for welcoming me in your lab: Tarun, Ahmed, Karim and Samantha.

Je remercie tous mes collègues du LIP6, les permanents (avec une pensée pour Marguerite) et les doctorants, en commençant par ceux qui étaient déjà au LIP6 quand j'ai commencé et qui sont partis (ou presque) : Tiphaine, Alexandru, Filippo, Marco, Jordan, Ahlem, Fadwa, Raul, Thiago et Matteo (félicitations !) ; ceux qui ont commencé avec moi et qui finissent maintenant (ou presque) : Alexandre (depuis la L1 !), Quentin, Giulio, Davide, Antonella, Rudyar et Loïc ; et enfin ceux qui vont rester (bon courage !) : Minh, Mustafa, Narcisse, Florian, Amr et Salah.

Je tenais aussi à remercier mes amis qui, depuis longtemps, ont subi mes monologues un peu trop *nerd* ou *geek* à leur goût. Donc, merci à Christophe, Sébastien, Baptiste, Cyril, Alexis, Laurent, Hubert et Philippe.

Enfin, je voulais remercier ma maman, mon papa, ma soeur, Djeamson, et surtout Nohemi qui ne m'a pas seulement supporté tout au long de cette thèse, mais qui m'a aussi beaucoup *supported* (hehe) dans cette aventure. And for this, I am super grateful!

Résumé de la thèse

Dans cette thèse, nous appréhendons la mobilité sous un nouvel angle. Nous proposons d'exploiter la mobilité quotidienne articulée autour de véhicules pour créer un médium de communication ad hoc. Notre objectif est de tirer partie des trajets quotidiens effectués en voiture ou en transport en commun pour surmonter les limitations des réseaux de données tels que l'Internet. Dans une première partie, nous tirons partie de la bande passante que génèrent les déplacements de véhicules équipés de capacités de stockage afin de délester (*offload*) en masse l'Internet d'une partie de son trafic. Les données sont détournées vers des équipements de stockage appelés points de délestage installés aux abords de zones où les véhicules s'arrêtent habituellement. Ces équipements agissent comme des points relais où les données sont entreposées avant d'être chargées et transportées sur un véhicule, et ce jusqu'au prochain point de délestage où elles pourront éventuellement être déchargées. La sélection des véhicules alloués au transport des données dépend de leur itinéraire mais aussi des contraintes qui caractérisent les données délestées. Nous formulons le problème de l'allocation des véhicules sous forme d'un modèle de programmation linéaire qui maximise le débit total résultant du transport des données sur le réseau routier tout en garantissant leur équité. Pour résoudre ce modèle, nous virtualisons les flots de véhicules circulant sur les segments de route reliant les points de délestage. Le réseau virtuel résultant permet de caractériser la mobilité en termes de bande passante, de délai et de taux de perte tout en réduisant la complexité de l'infrastructure de délestage. Nous proposons ensuite d'étendre le concept de point de délestage selon deux directions dans le contexte de services reposant toujours la mobilité des véhicules. Dans la première extension, nous proposons d'exploiter les capacités de stockage des points de délestage afin de concevoir un service de stockage et partage de fichiers offert aux passagers de véhicules. Les points de délestage se comportent comme des points de stockage où les passagers déposent leurs fichiers, qui sont ensuite répliqués au niveau des autres points de stockage en utilisant le mouvement des véhicules. Afin d'exécuter les requêtes des passagers pour déposer ou récupérer un fichier dans des délais contraints, nous proposons un algorithme qui détermine les emplacements des points de stockage de telle façon que chacun capture un nombre maximum de requêtes de passagers avant qu'elles n'expirent. Dans la seconde extension, nous dématérialisons les points de délestage en zones géographiques pré-définies. Ces zones se distinguent par leur concentration d'un grand nombre de véhicules où les communications entre véhicules durent suffisamment longtemps pour transférer de grandes quantités de données. L'évaluation des performances des différents travaux menés au cours de cette thèse montrent que la mobilité inhérente aux entités du quotidien permet la fourniture de services innovants avec une dépendance limitée vis-à-vis des réseaux de données traditionnels.

Mots-clefs

Délestage de données, réseaux véhiculaires, gestion de ressources, virtualisation, *software-defined networking*, *delay-tolerant networking*.

Thesis abstract

In this thesis, we tackle mobility from a new perspective. We exploit the daily mobility of vehicles to create an alternative transmission medium. Our objective is to draw on the many vehicular trips taken by cars or public transports to overcome the limitations of conventional data networks such as the Internet. In the first part, we take advantage of the bandwidth resulting from the mobility of vehicles equipped with storage capabilities to offload large amounts of delay-tolerant traffic from the Internet. Data is transloaded to data storage devices we refer to as *offloading spots*, located where vehicles stop often and long enough to transfer large amounts of data. Those devices act as data relays, *i.e.*, they store data it is until loaded on and carried by a vehicle to the next offloading spot where it can be dropped off for later pick-up and delivery by another vehicle. The vehicles allocated to the transport of data are selected according to their direction so as to meet the performance requirements of data transfers. We formulate the data transfer allocation problem as a linear programming model, which maximizes the cumulative throughput achieved by the transfers of offloaded data in a fair manner. To solve this model, we characterize the flows of vehicles traveling the road segments connecting adjacent offloading spots into network quantities such as capacity, delay, and loss. The resulting overlay network reduces the complexity of the road network topology and makes the data transfer allocation tractable. With actual traffic counts of the French roads, we show that the road network has the potential to offload several Petabytes of data per day. We further extend the concept of offloading spots according to two directions in the context of vehicular cloud services. In the first extension, we exploit the storage capabilities of the offloading spots to design a cloud-like storage and sharing system for vehicle passengers. The offloading spots act as repositories where users first upload their files, which are then replicated among the other repositories using the movements of the vehicles traveling between the repositories. To process the requests to store or retrieve a file in a timely fashion, we design a placement algorithm that determines the locations where the repositories can capture a maximum number of user requests before they expire. In the second extension, we dematerialize the offloading spots into pre-defined areas that feature high densities of vehicles and where vehicle-to-vehicle communications last long enough to transfer large amounts of data. We exploit these areas to migrate virtual machines between vehicles without involving intermediate offloading spots. The performance evaluation of the various works conducted in this thesis shows that everyday mobility of entities surrounding us enables innovative services with limited reliance on conventional data networks.

Keywords

Offloading, vehicular networks, resource management, virtualization, software-defined networking, delay-tolerant networking.

This page intentionally left blank

Contents

This page intentionally left blank

1 Introduction

1.1 | Context

The increased popularity of data-intensive applications drives content providers to create new approaches to replicate and synchronize large amounts of data across geo-distributed data centers. In this context, *traffic offloading* is gaining interest, as it represents a cost-effective solution to *extend* network capacity. Offloading involves exploiting alternative transmission media and data delivery models [Pat03].

Well-known content providers leverage offloading techniques to deliver content. For instance, in 1998, Netflix began offering a movie rental service to its customers, enabling them to receive DVD in the mail.[1] Amazon has developed the Import/Export service, which allows Amazon Web Services (AWS) customers to ship hard drives to Amazon's data centers.[2] This service is particularly useful for customers limited by slow Internet connections. More recently, AWS launched Amazon Snowball, a portable ready-to-be-shipped appliance with a storage capacity of 50 TB that can accommodate transfers of several Petabytes to AWS. These offloading techniques intend to overcome the challenges resulting from large-scale data transfers such as high network costs, long transfer times, and security concerns.

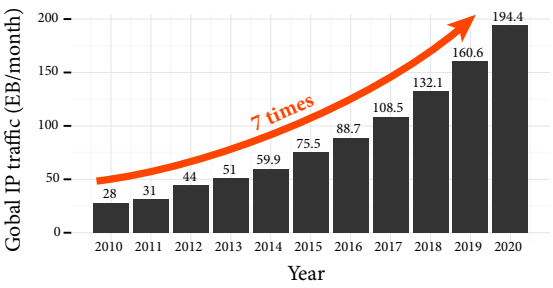


Figure 1.1. Global IP traffic [Cis16].

In this thesis, we propose to equip private vehicles with storage devices with the purpose of turning the road network into a large capacity transmission system. A back-of-the-envelope calculation shows that 10% of the vehicles traveling the roads of France equipped with a 1 TB hard drive can transport up to 115 EB per day (1.3 PB per second). Extending this idea to the 1.2 billion cars available worldwide, the everyday mobility of vehicles represents an untapped potential for addressing the oncoming data tsunami. With data-intensive applications such as high-definition video, autonomous vehicles, and virtual reality, the demand for network bandwidth grows by an estimated 22% per year. Already the case in some situations, in the next few years, the data consumption will outpace the network improvements made by providers in laying high-capacity fiber and speeding up their already-existing network infrastructure and servers [Hec16]. As shown in Figure 1.1, the demand for bandwidth is at its strongest today:

[1] `http://dvd.netflix.com/`

[2] `https://aws.amazon.com/fr/importexport/`

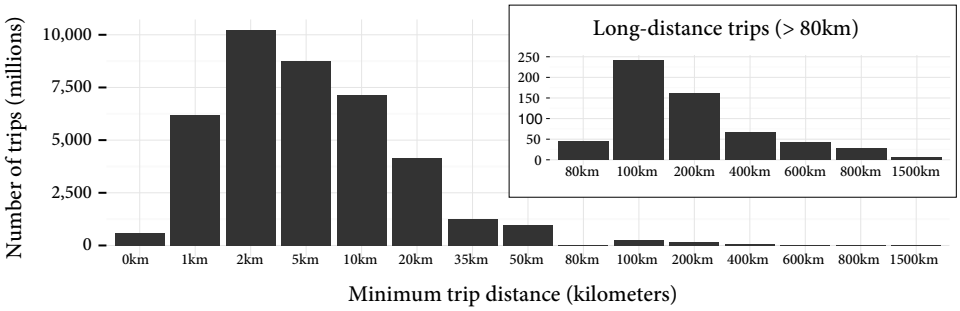


Figure 1.2. Distribution of trips by distance travelled by vehicles in France. Source: the French National Household Travel Survey (ENTD), 2008 [MED].

global IP traffic has grown threefold during the past six years and is expected to double within the five next years [Cis16, GR12].

The opportunistic use of vehicles follows the current trend of services that take advantage of the sharing economy (or access economy), such as BlaBlaCar and Uber, by enabling peer-to-peer-based sharing of goods and services. In our case, *the trips made by the vehicles are shared resources we use to transport data on behalf of content providers*.

Equipping vehicles with data storage also follows the second trend which consists in building value-added services right into an existing business. Examples of this trend include parcel delivery services built on top of existing passenger transportation services. In particular, Greyhound offers such a service that "piggybacks" on the trips made by buses as a part of the coach service.[3] In the same manner, Uber proposes UberRUSH, an on-demand delivery service that relies on the fleet of Uber vehicles normally dedicated to passengers transportation.[4] In our case, private vehicles transport data for the account of content providers in exchange for their normal routine (*e.g.*, to synchronize backup data between remote data centers they operate). A service provider is then in charge of supervising the offloaded transfers then charged to the content providers and providing incentives to the vehicle owners for the data they transport (*e.g.*, through a "get paid to drive" program).

> *In this thesis, we characterize and exploit the existing mobility of everyday entities to mitigate or remove the limitations of conventional data networks. We rethink the design of popular services such as bulk data transfers or cloud-like file sharing by limiting or removing their reliance on such networks.*

1.2 | *Vision and data offloading design*

We exploit the *delay-tolerance* of background traffic to offload bulk data transfers from a conventional data network such as the Internet over the road network. We target data transfers in

[3] `http://www.shipgreyhound.com/`

[4] `https://rush.uber.com`

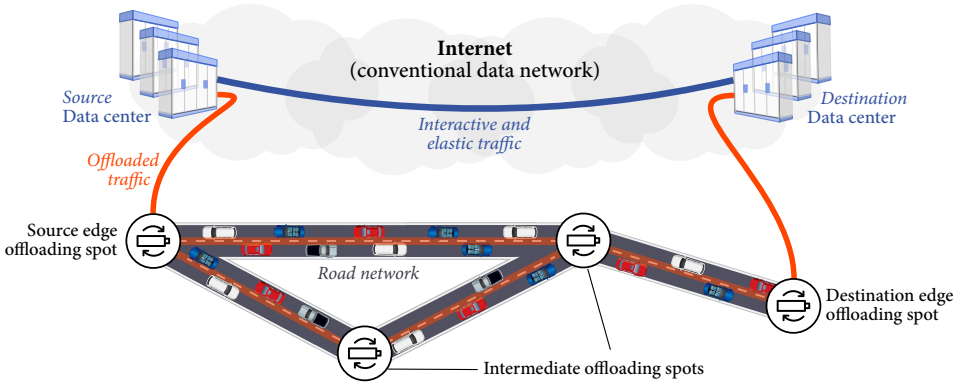


Figure 1.3. General overview of the data offloading we propose in this thesis.

the context of applications with a delay tolerance of several days (*e.g.*, distribution of large scientific datasets or traffic resulting from maintenance and provisioning activities of large-scale distributed systems) [LSRS09]. As noted by Hecht, the synchronization of the private content between remote data centers is one of the biggest drivers of bandwidth demand [Hec16]. Our data offloading takes opportunistic advantage of the increasing number of vehicular trips to physically transport data between locations [LJ10].

One simple realization of the data offloading is to rely on the vehicles making the trip all the way from the source to the destination of a data transfer. However, as shown in Figure 1.2, less than 1.4% of the trips made by vehicles in France are longer than 80 km. While this number is still significant for short-distance data transfers (it corresponds to about 600 million trips per year), it becomes very low when the distance increases. Moreover, the number of destinations also increases with the distance of the trip, which results in just a few trips to transfer data over large distances. An alternative consists in using a dedicated fleet of vehicles acting as *ferries* to transport the data from its source directly to its destination. However, this solution is costly and does not scale when the number of transfers increases, as more vehicles are needed to deliver the data.

Instead, we rely on dedicated facilities to *compose* several flows of storage-enabled vehicles traveling in different directions. We refer to these facilities as *offloading spots* as we depict in Figure 1.3. Offloading spots are equipped with dedicated storage to temporarily store data and pass it between vehicles belonging to different flows. They are placed where vehicles stop often and long enough to transfer large amounts of data. Examples of such locations are on-street parking spots, garage parking, gas and electric charging stations, or supermarket parking lots.

Following up on the work on traffic offloading, we extend the concept of offloading spot according to two distinct directions both in the context of vehicular cloud services. In the first extension, we leverage the storage capacity of the offloading spots to turn them into repositories and provide a geo-distributed system to store and share files of mobile users. To enforce the file sharing, we rely on the movements of vehicles traveling between the offloading spots to

replicate the files and make them accessible to the mobile users. In the second extension, we virtualize the resources of the vehicles to create a virtual vehicular network. We dematerialize the offloading spots into pre-defined areas that feature high densities of vehicles to enable transfers of virtual machines between the vehicles. With these extensions, we further show that the existing mobility of vehicles has the potential to provide cloud services with limited reliance on conventional data networks.

1.3 | *Challenges*

Firstly, we need to efficiently allocate data transfer demands to the existing flows of vehicles. We refer to this problem as the *vehicle flow allocation problem*. To solve it, we need to design an efficient allocation process of the data transfers that optimizes target performance metrics, while matching the performance requirements of the corresponding demands.

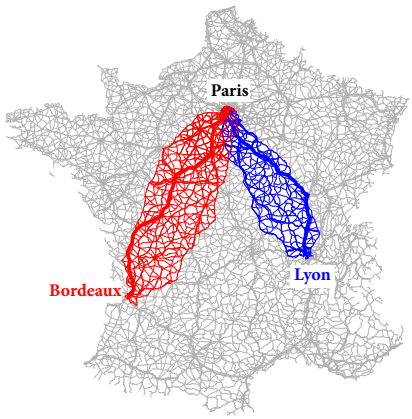


Figure 1.4. Main roads connecting Paris to Lyon and Bordeaux.

Secondly, we need to cope with the scale of the road network in order to achieve a scalable and efficient vehicular data offloading. The complexity of the road network's topology and the large number of vehicular trips both make the vehicle flow allocation problem computationally intractable. We illustrate the need for a scalable allocation mechanism in Figure 1.4, where we depict the routes of the potential trips that can be allocated to offload transfers originating from Paris to Bordeaux and Lyon.

Finally, we need to ensure reliable data transfers. Since vehicles may fail to deliver the data they carry to the next offloading spot or the final destination, we need to rely on and adapt techniques to recover from the data losses.

In the extensions of our main work, the underlying challenge is to determine the locations of the offloading spots, whether they are materialized or not, according to the specific requirements of the services we deploy.

In our vehicular file sharing system, we need to design an algorithm to place the repositories at strategic locations where they can capture a maximum number of user requests before they expire, while connecting them together by the movements of the users to enable the replication. In our virtual vehicular network, we need to identify the location of the areas where vehicles meet frequently and long enough to transfer large amounts of data, such as virtual machines.

1.4 | *Contributions and thesis outline*

First contribution: Survey and taxonomy of techniques to transfer data using everyday mobility (Chapter 2). We begin this thesis by providing an exhaustive survey and taxonomy of

the existing techniques that help exploit the mobility of everyday entities to transport data in the context of various applications. We categorize the existing techniques based on whether the delivery involves a single entity or multiple entities that take turns to transport the data. We also describe the various approaches proposed in the literature for characterizing the mobility and controlling the data forwarding.

Related paper:

- Benjamin Baron, Prométhée Spathis, Marcelo Dias de Amorim, and Yannis Viniotis. "Data transfers using everyday mobility: A survey." Submitted to *IEEE Communications Surveys & Tutorials*, 2016.

Second contribution: Assessment of the vehicular offloading concept (Chapter 3). In Chapter 3, we assess the concept of vehicular data offloading. We compare the cost of transferring data using the existing mobility of vehicles to the cost of transferring the same amount data on the Internet. We first introduce a map reduction procedure which mitigates the complexity of the road network. The output of this procedure is a logical representation which translates and characterizes the flows of vehicles into network quantities. This representation is used to solve an allocation procedure which selects the flows of vehicles in charge of carrying the offloaded data. We formulate and solve this allocation procedure as a *revenue maximization model* that allocates the flows of vehicles so as to maximize the cost-benefit of offloading data on the road network compared to data transfers over conventional data networks. Finally, we show using actual traffic counts for the French road network that our offloading concept can achieve an aggregate capacity in the Petabyte range per week.

Related papers:

- Benjamin Baron, Prométhée Spathis, Hervé Rivano, and Marcelo Dias de Amorim. "Offloading massive data onto passenger vehicles: Topology simplification and traffic assignment." *IEEE/ACM Transactions on Networking*, 2015.
- Benjamin Baron, Prométhée Spathis, Hervé Rivano, and Marcelo Dias de Amorim. "Vehicles as big data carriers: Road map space reduction and efficient data assignment." *IEEE Vehicular Technology Conference (VTC Fall)*, 2014.
- Benjamin Baron, Prométhée Spathis, Hervé Rivano, and Marcelo Dias de Amorim. "Étude de l'intermodalité pour le délestage des réseaux d'infrastructure." *Algotel 2015 — 17èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications*, 2015.

Third contribution: Offloading infrastructure centralized control (Chapter 4). We propose an architecture that implements the concept of vehicular data offloading. We rely on a centralized architecture that enables efficient control of the offloading infrastructure. This architecture consists of a controller and the collection of offloading spots acting as forwarding engines. The controller is in charge of selecting the road network path for each offloading demand it receives. The selection of road network paths is done so as to maximize the throughput of the resulting transfer while ensuring the fair allocation of flows of vehicles traveling the paths among the other transfers that allocated to the same paths. This is achieved by modeling and solving the allocation of a data transfer to a road path according to a max-min fairness problem. The

controller translates the output of the allocation of a data transfer by a set of forwarding rules installed at the offloading spots composing the selected road network. Finally, the controller ensures the reliability of the offloaded data using redundancy and retransmission techniques. With simulations on the French road network and elaborated traffic modelling techniques presented in the Appendix A, we show that the offloading architecture we propose has the potential to offload several Petabyte of data per day.

Related papers:

- Benjamin Baron, Prométhée Spathis, Hervé Rivano, Marcelo Dias de Amorim, Yannis Viniotis, and Mostafa Ammar. "Data Haulage: Efficient resource management on the road network." Submitted to *IEEE Transactions on Network and Service Management*, 2016.
- Benjamin Baron, Prométhée Spathis, Hervé Rivano, Marcelo Dias de Amorim, Yannis Viniotis, and Joseph Clarke. "Software-defined vehicular backhaul." *IFIP Wireless Days (WD)*, 2014.

Fourth contribution: Vehicular cloud services (Chapter 5). We propose two extensions to the work we conduct in traffic offloading. We extend the concept of offloading spots in the context of two vehicular cloud services. A first service turns the offloading spots as repositories where mobile users can upload and share files. We propose a placement algorithm derived from the Maximal Covering Location Problem [CV74] that determines the location of a target number of repositories such that (*i*) a maximum of users are covered before their uploading or retrieval requests expire and such that (*ii*) the synchronisation of the repositories can be done by using the existing movements of the users.

The second service virtualizes the resources of vehicles operating within a large mobile network. The offloading spots refer to specific areas where the virtual machines exploiting the abstract representation of the vehicle resources can migrate between vehicles while in contact. We propose a methodology for analyzing the spatial distribution of the contact density. This methodology determines the specific areas where contacts are long and frequent enough to accommodate the migration of virtual migrations.

Related papers:

- Benjamin Baron, Miguel Campista, Prométhée Spathis, Luís Henrique MK Costa, Marcelo Dias de Amorim, Otto Carlos MB Duarte, Guy Pujolle, and Yannis Viniotis. "Virtualizing vehicular node resources: Feasibility study of virtual machine migration." *Elsevier Vehicular Communications*, 2016.
- Benjamin Baron, Prométhée Spathis, Marcelo Dias de Amorim, and Mostafa Ammar. "Cloud storage for mobile users using pre-positioned storage facilities." *ACM SmartObjects*, 2016.

Thesis outline. This manuscript is organized in six chapters, including this introduction (Chapter 1). Chapter 2 presents a review of previous approaches that propose to use mobility of entities to transport data. The two following chapters 3 and 4 present the main contribution of

the thesis with, respectively, a revenue analysis and a practical implementation of the vehicular data offloading system. Chapter 5 presents two extensions of the data offloading system for vehicular cloud services. Each chapter begins with a brief introduction followed by a list of the main contributions. Given the novelty of our work, we provide additional discussions for each chapter on related topics we chose not to address in detail, including data security, privacy concerns, and road traffic dynamics. Finally, we conclude this thesis with a last chapter (Chapter 6) that summarises our work and a raises futher questions as a plan for future research.

Collaborations. The research presented in this thesis was conducted in the context of a number of scientific collaborations: Hervé Rivano (Inria), Yannis Viniotis (NCSU and Cisco), Joe Clarke (Cisco), Miguel Campista (UFRJ), Luís Henrique MK Costa (UFRJ), Otto Duarte (UFRJ), Guy Pujolle (UPMC), and Mostafa Ammar (Georgia Tech).

2 Data transfers using everyday mobility: Taxonomy and state of the art

The emergence of wireless capabilities equipping an ever-growing range of mobile entities has led to various paradigms such as Mobile Ad hoc Networks (MANETs), Vehicular Ad hoc Networks (VANETs), Wireless Sensor Networks (WSNs), and Delay-Tolerant Networks (DTNs). These paradigms enable the connectivity of a wide range of mobile entities in the context of various application scenarios ranging from transportation systems to sensing platforms.

These different scenarios exploit entities that are mobile either by nature, such as humans or animals, or by conception, like vehicles and robots moving in challenging environments. The entities are equipped with storage capabilities, allowing them to store and carry data. They are also equipped with communication interfaces to forward data when the entities come in *contact*, *i.e.*, when they are in each others' communication ranges. While most of the approaches propose forwarding strategies to cope with the mobility of the entities, we focus on those that exploit the carrying phase resulting from the movements of the mobile entities.

In this chapter, we review and provide a comprehensive categorization of the *existing strategies that leverage the existing mobility of entities to physically move data*. The movements of the entities provide an alternative transmission medium, as illustrated in Figure 2.1.

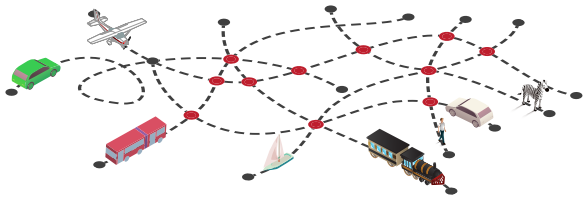


Figure 2.1. Alternative transmission medium based on the movements of entities.

The idea of exploiting the mobility of different entities has been proposed in the context of various applications that do not require real-time communications. Some consider there is no data network available and entities are in charge of bringing network connectivity. Mobile entities can also be used in tandem with a conventional data network to extend the network coverage or capacity. In general, the applications rely on the following performance metrics to measure the efficiency of the data delivery:

- *Delivery delay* — the average delay to deliver the data from the sources to the destinations.
- *Delivery rate* — the amount of data delivered to destinations per time unit.
- *Delivery ratio* — the amount of data delivered to destinations divided by the total amount of data sent by sources that has been delivered.
- *Energy efficiency* — the average amount of data delivered per unit energy consumption.

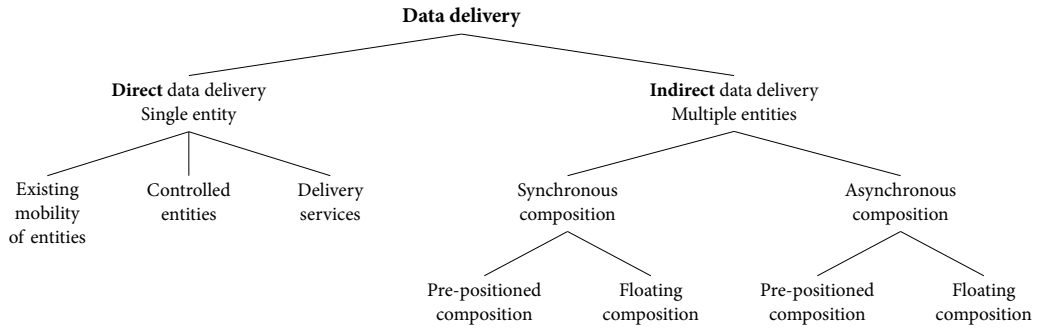


Figure 2.2. Classification of the trajectory composition according to the type of data delivery.

We represent the structure of the classification in Figure 2.2. Some works propose the use of a single entity to carry the data from its source all the way to the final destination. This results in a *direct data delivery*, which we present in Section 2.1. However, relying on a single entity is somehow limiting from the perspective of the expected benefits (*e.g.*, capacity or coverage). The idea of composing the trajectories of multiple mobile entities was thus introduced. According to this idea, the data follows a path consisting of multiple segments of trajectories, each followed by different mobile entities. This results in an *indirect data delivery*, which we present in Section 2.2. Data is passed from one entity to another as a result of the forwarding when entities are in direct contact.

In the case of indirect data delivery, we choose to present the works that are the most relevant to this thesis. They propose different strategies for composing the trajectories of mobile entities. These strategies differ depending on the two following criteria:

- The *time* when composition should occur. The data is passed either *synchronously* when two entities meet or buffered before passed *asynchronously* to subsequent entities.
- The *location* where composition is performed. The location can be either *pre-defined* or *floating*. In the pre-defined case, the data is passed if the entities are in contact at specific locations. In the floating case, composition results of contacts between entities regardless where they are in contact.

The strategies proposed in the literature result of the combination of one of the instances of both previous criteria.

2.1 | Direct data delivery

In this section, we present the direct delivery strategies that exploit the movements of a single entity to carry data. We classify these strategies according to the degree of knowledge regarding the mobility of the carrying entity.

Firstly, we consider the strategies using entities exhibiting random mobility. Although they eventually deliver the data they carry, the delivery delay can be excessively long, making the transfers unreliable and unfit for real-life applications.

Having partial knowledge about the movements of the entities allows the deployment of applications that require performance guarantees, such as sensing platforms and systems to extend the connectivity of remote areas.

Secondly, we consider the entities whose mobility is pre-calculated in advance. The related strategies and the related approaches to control their movements to transport the data. Controlling the movements of the entities provides better guarantees on the delivery delay and rate, which matches the requirements of sensing applications.

Finally, we review the approaches that leverage third-party services such as postal services to transport data. These services allow the transfer of large amounts of data, resulting in a large capacity alternative medium for large-scale bulk data transfers.

2.1.1. Existing mobility of entities

Several approaches rely on the existing movements of a single entity to transport the data in the context of delay-tolerant applications such as sensing platforms or periodic data transfers to bridge connectivity gaps in remote areas not covered by conventional data networks. The data is loaded in the storage of the mobile entity when it comes close to the source (*e.g.*, sensor nodes or gateway connected to a data network). As represented in Figure 2.3, the entity transports the data as part of its usual movements and unloads the data when it comes close to the destination (*e.g.*, sensor sink or gateway). Since only one copy of the data is carried, this approach does not consume many resources and has a minimum overhead. With no knowledge on the mobility of the entity, there are no guarantees on the expected delivery, which makes this strategy highly unreliable. However, some approaches exploit the knowledge of the mobility of the entity to deploy applications with stronger performance guarantees.

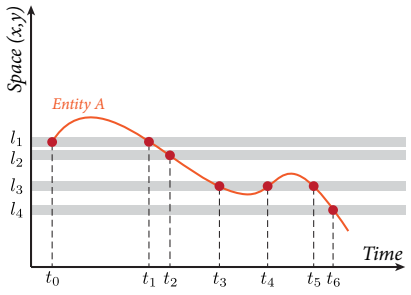


Figure 2.3. Direct delivery using the existing movements of a single entity.

Random entity mobility. In the case of a random mobility, Grossglauer and Tse show how to use entities with unknown mobility to transport data instead of using traditional ad hoc networks where a wireless multi-hop path must always exist between the source and the destination [GT01]. The direct delivery approach leads to better throughputs under large data loads compared to the ad hoc case, as most of the traffic carried by the entities in ad hoc networks is relayed traffic, which consumes both energy and bandwidth. However, using the mobility of the entities leads to high delays, as the entity carrying the data may take a long time before coming in contact with the destination.

Large-scale sensing platforms. Large-scale sensing platforms can benefit from the random movements of entities to collect data from sensors scattered in an area, or even generate environmental data as part of their movements when equipped with sensors. The data is temporarily stored and carried by the entities and delivered to a central sensor sink when the entity moves close. The sensor sink aggregates the data to transfer it to remote servers using a conventional

data network for long-term storage and analysis. Compared to ad hoc networks, this transmission model allows large power savings and higher bandwidth at the sensor nodes because the communications take place over short ranges, which reduces the power of the radios.

Burrell *et al.* study the deployment of sensor networks in vineyards to take effective decisions in the face of low temperatures, rain or frost. The authors relied on vineyard workers to transport the data from the sensors to the sink located in the farm [BBB04]. SeNDT (Sensor Networking with Delay Tolerance) is a system for water pollution and noise monitoring at a lake in the center of the Republic of Ireland, which relies on sensors deployed around the lake [MGH+07]. To collect the data, the authors mounted wireless devices on anglers' boats that they use to fish in the lake to collect sensor data as part of their movements on the lake. The data is then aggregated at a sink in the boathouse when the anglers return. The EMMA project aims at monitoring pollution using buses to generate location-based environmental data along their movements [LDP+07]. As part of their routes, the buses transport the data to stationary sinks placed in strategic locations, such as intersections. Using the existing movements of entities as part of sensing platforms to generate or collect sensing data avoids costly deployments to transmit the data through intermediate nodes and increases the lifetime of the sensor nodes by using short-range radios. However, such platforms must have loose constraints on the delivery rate and delay of the data as they rely on unpredictable movements of the entities.

With knowledge on the mobility patterns of the entities, DakNet leverages the bus schedules to provide Internet connectivity to remote regions of India and Cambodia [PFH04]. Equipped with mobile access points, the buses physically transport the data between Internet access points located in larger cities and rural kiosks located in the villages to exchange delay-tolerant data such as mail or land records. This provides an efficient low-cost alternative to expensive dialup landlines or long-range radios. Compared to the previous approach, DakNet gives better guarantees because of the fixed schedule of the buses. A similar initiative was proposed by the Wizzy Digital Courier service to provide connectivity to access schools located in remote villages of South Africa [Rab04].

Figure 2.4. A pigeon equipped with a backpack carrying an SD card.

RFC 1149 specifies an experimental method to transmit IP datagrams using avian carriers, as represented in Figure 2.4 [Wai90]. Although suited for very specific scenarios, this solution becomes infeasible in large-scale networks due to the high delay and low throughput of the transfer. Some implementations of the protocol lead to large losses (with over 55% of lost datagrams) and variable delays of several hours when transporting the data over five kilometers.

While the endpoints of the pigeon route are pre-defined, its movements are random and lead to losses, making the transfer unreliable and unsuitable for the targeted applications. In the following, we overview the different approaches that leverage controllable entities to provide reliable data transport.

2.1.2. Controlled entities

Contrary to using the existing movements of entities, controlling their movements provides better performance guarantees, for instance on the delivery rate and delay of the data. Generally referred as "ferries" or "robots", these dedicated entities are generally used in sensor networks to provide communication capacity among the sensor nodes by picking up data from the source(s) and carrying it to the destination(s), as depicted in Figure 2.5. In this figure, a ferry periodically visit the nodes located at l_1, l_2, l_3, and l_4. An underlying challenge is to compute a route for the dedicated entity such that it visits all the nodes and optimizes performance metrics (*e.g.*, average delivery delay and rate). In the work we review, the nodes generate data traffic (uniform or not), which results in different bandwidth requirements that the dedicated entity has to satisfy. Otherwise, the data generated at the nodes accumulates to a point where it is dropped in case of limited buffer capacity.

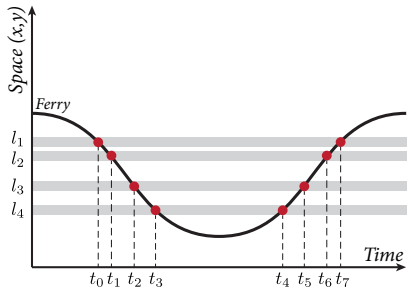


Figure 2.5. Direct data delivery with a controllable entity.

Message ferry. In the initial message ferry work, Zhao *et al.* proposed an algorithm to compute the ferry route of a single ferry such that it minimizes the average delivery delay of the data generated uniformly by a collection of stationary nodes [ZA03]. The algorithm first creates a route using local optimization techniques for the Travel Salesman Problem (TSP) (2H-opt) that minimizes the average delay for all traffic generated by the nodes. Second, the algorithm extends the original route to meet the bandwidth requirements of the nodes. While this technique works well with equal node bandwidth requirements, it does not adapt to unequal ones.

Unequal bandwidth requirements at the entities. With unequal bandwidth requirements, Mansy *et al.* propose to use the deficit round-robin technique to decide which node to visit next depending on the visit history and the last meeting with the ferry [MAZ11]. With this approach, nodes with higher data rates get better chances to be visited by the ferry. Tirta *et al.* propose to decrease the delay of the ferry route by visiting fewer nodes using the ad hoc connectivity of the nodes to form clusters [TLLB04]. The ferry only visits one node per cluster, the *cluster head*, which aggregates and distributes the data from and to the other nodes of the cluster. The authors focus on the impact of different route schedules on the buffer size of the cluster head, which sees a large load of messages to aggregate and distribute. The authors show that scheduling the visits of the ferry to minimizing its movements reduces the energy consumption of the ferry. However, scheduling the visits of the ferry at the cluster heads according to the data rate aggregated at the cluster heads leads to better delivery delays.

Multiple controllable entities. Adding more controllable entities to collect the data of the nodes increases the cost, but also improves the network capacity and overall performance, as the entities can cover larger distances, handle more traffic load, and provide better reliability in case of failure. Zhao *et al.* propose two different techniques to directly deliver data among the nodes using multiple message ferries [ZAZ05]. The first technique consists in designing a single route using the same heuristic as followed by all the ferries with the same speed and different

timings, allowing each node to communicate with all the ferries [ZA03]. The second technique is more complex and consists in designing multiple routes followed by the ferries instead of a single route. Since the data is not relayed by the ferries, each route must accommodate all the bandwidth requirements of the node visited by the ferry on the route. The authors propose a greedy heuristic that assigns nodes to ferries and balances the traffic load among the routes by shortening or extending them such that the estimated weighted delay of the data traffic among the nodes is minimized.

More recently, pigeon networks introduced *pigeons* that act as ferries between a "home host" and "foreign hosts" on dedicated routes [ZRL+13]. There are no data exchanges between the ferries, so each route must accommodate the requirements of every foreign host they visit. The authors propose an exact optimization technique to design the routes of the pigeons such that they minimizes the average delay of the generated data. This technique is optimal for small-scale networks but becomes intractable for larger networks. To make the problem tractable, they propose a geographical partitioning-based heuristic that relies on the divide-and-conquer idea to design the pigeon route. The authors show that the resulting route yields lower delays compared to the non-partitioning strategy used in the route design of the message ferry work.

Real-world deployments. Several projects have studied and deployed controllable entities in real-life settings in the context of sensing platforms. Tekdas *et al.* deployed a small sensor network with robots to measure the energy savings brought by the addition of the robots [TILT09]. With robots assigned to different routes using the k-TSP heuristic, which finds k tours where the largest of the k tours is minimized, they showed significant energy savings at the sensor nodes from 1mW in tradition ad hoc to 0.003mW with the robots. Since the robots come close to the sensor nodes, they require limited transmission power and fewer transmission attempts, which decreases the overall energy consumption.

Underwater networks. Underwater networks of sensors deployed in the oceans for long-term environmental monitoring or surveillance also leverage robots to collect sensor data [PKL07, APM05, DCVR06]. Because of the cost of the underwater sensor nodes and the extent of the areas to monitor, the deployment of underwater sensor networks is generally sparse, making it impossible to form an unpartitioned ad hoc network. Robots or Autonomous Underwater Vehicles (AUVs) transport sensor data between the sensor nodes in the partitions. These networks differ from terrestrial sensor networks because of the acoustic channel shared between the communications among the sensors and the AUVs, and the navigation system of the AUVs. This twofold utilization of the medium creates contentions, limiting its use. The AUVs use visual odometry to travel to the next sensor node and collect its data. The AUVs then send via a high-bitrate channel the data collected from the underwater sensors to buoys acting as sinks.

Using controllable entities to enable the communication between static nodes gives better delivery delays and rates, as the route followed by the entity is specifically designed to optimize these metrics. In the following, we review the approaches that use existing delivery services to transport the data instead of controlling the mobility of entities. These approaches rely on entities already operated and managed by the delivery services.

2.1.3. Delivery services

Delivery services (*e.g.*, USPS, La Poste, FedEx, or UPS) provide an alternative mean to transport data by shipping hard drives and disks over long distances, as they avoid deploying and maintaining costly conventional data networks. Delivery services target delay-tolerant transfers and provide guarantees on the delivery date and shipping cost of the parcels. We represent their use in Figure 2.6 where the data is transported from location l_1 to location l_2.

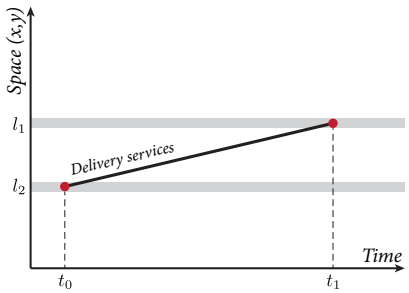


Figure 2.6. Direct data delivery using delivery services.

Wang *et al.* proposed Postmanet, a generic system that complements the Internet and provides connectivity to areas in rural regions using postal services [WSG+04]. The data is copied on DVDs and shipped in parcels to remote Postmanet routers that act as gateways similar to rural kiosks, which, in turn, put the received data at disposal to the remote users (*e.g.*, email or land records). The authors overview different routing strategies to ship the data, including direct shipping between users. Since the other strategies rely on intermediate servers to dispatch the data, we will review them in Section 2.2.2.2 where we review the indirect delivery approaches.

Figure 2.7. Amazon Snowball.

Laoutaris *et al.* compared the cost of sending data using postal services with the cost of sending the same amount of data through a conventional data network, as content providers operating several datacenters would do [LSRS09, LSS+13]. Such data transfers are currently serviced by expensive dedicated networks (*e.g.*, Large Hardon Collider Computing Grid) or by postal services. The authors exploit intermediary storage in transit Internet Service Providers (ISPs) to temporarily buffer the data and send it during off-peak hours when the costs to transmit data are low. The authors found that it is less expensive to use the postal services for individual shipments that occasionally happen (*e.g.*, short-lived transfers). However, in the case of a constant flow of data to transfer, they show that shipping data is more expensive than sending it through an ISP using the intermediate storage the authors propose to use.

Commercial applications using delivery services. Many commercial applications rely on postal services to exchange content with their customers. As an example, Netflix offers a DVD rental service to its customers[1] which allows them to rent DVDs and receive them in the mail. Netflix' customers may keep a DVD as long as they want, and must return it before renting another DVD. With this service, Netflix currently distributes more than 1.5 million DVDs a day[2], accounting for a total aggregate bandwidth of more than 650 Gbps.

Large cloud providers offer services to allow their customers to ship physical media of their data to be uploaded to remote cloud storage (*e.g.*, AWS Import/Export[3], Microsoft Import/Export

[1] http://dvd.netflix.com/
[2] http://ir.netflix.com
[3] http://aws.amazon.com/importexport

Service[4] and more recently Google Offline Disk Import[5]). AWS launched the Amazon Snowball (shown in Figure 2.7) is a portable ready-to-be-shipped appliance with a storage capacity of 50 TB that aims to accommodate transfers of several Petabytes (using multiple Snowball) to AWS servers. Amazon compares this solution with transfers on common Internet connections and shows that it takes approximately a week to transfer 50 TB via a 1G connection, or to ship a Snowball with the same amount of data. Therefore, this solution exemplifies the potential of physically transferring data for massive data offloading.

2.2 | *Indirect data delivery*

As we have seen in the previous section, relying on the existing movements of a single entity to transport the data limits the expected performance of the resulting transfers, in terms of the delivery delay and rate. Instead of one entity, one can *compose* the trajectories of multiple entities to transport the data to its destination. Composing the trajectories of entities gives the opportunity to route the data on multiple paths, thus increasing the resulting capacity and reliability of the data transfers. As we have mentioned in the introduction of this chapter, we classify the different strategies for composing the trajectories of entities depending on the time and location of the composition.

Firstly, we review the strategies that rely on the synchronous composition of the trajectories of entities. In this case, the data is then passed from one entity to another whenever two entities come in contact. The different strategies we review propose more of less elaborated control planes to decide whether to forward, replicate, or keep the data during a contact to optimize the performance of the resulting data transfers.

Secondly, we review the strategies that rely on a dedicated infrastructure that consists of either stationary or mobile nodes with controlled mobility to asynchronously compose the trajectories of the entities. This infrastructure mitigates the effects of the partial knowledge of the movements of the entities and allows efficient compositions of the movements to improve the performance of the data transfers, in terms of delivery delay and rate.

2.2.1. Synchronous composition of the trajectories

If the data is passed in a synchronous way, the two mobile entities involved in the composition of their trajectories are required to be in contact at the same time. As a result, the data follows a route that consists of the successive compositions of the trajectories of the entities that physically carry the data towards its destination. This approach is commonly referred to as *mobility-assisted forwarding*, or also as *store-carry-and-forward*. The work we review in this section proposes different control plane approaches to make the forwarding decisions over contacts. Instead of using straightforward decisions based on a local control plane, the approaches

[4] `http://azure.microsoft.com/en-us/documentation/articles/storage-import-export-service/`

[5] `https://cloud.google.com/storage/docs/offline-media-import-export`

that leverage a global control plane make more informed and efficient forwarding decisions to improve the performance of the data transfers.

While the contacts can occur anywhere, only few should be considered to compose the trajectories of the entities. Some strategies only consider the compositions that result of the contacts occurring in specific pre-defined areas. These areas correspond to locations with higher densities of entities with more contact opportunities than the other locations with lower densities. Routing data between these areas enables better control of the forwarding and requires fewer replications to deliver the data.

In the following, we present the strategies that transfer the data over synchronous compositions of the trajectories of entities either with floating compositions or at pre-defined locations.

2.2.1.1 Floating composition

When two entities come in contact at the same time and location, it creates an opportunity to compose their trajectories and bring the data closer (in time) to its destination. We represent the floating composition in Figure 2.8 where the data is passed from entity B to entity A. During a contact, the two entities can forward, replicate, or keep the data they carry. In this section, we review the different strategies to synchronously compose the trajectories of entities. The applications envisaged with these strategies must tolerate delayed deliveries. They range from sensor networks to vehicular networks and battlefield communications.

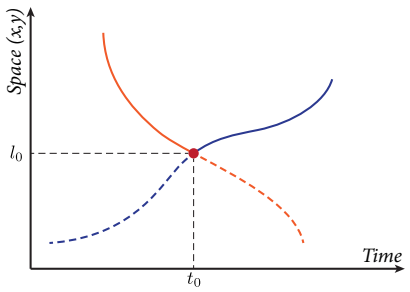


Figure 2.8. Indirect synchronous floating composition between two entities.

No control plane. With no control plane, the entities compose their trajectories with any other entities they encounter by replicating the data they carry. This is how the epidemic protocol works by disseminating the data [VB+00]. Every intermediate entity that receives the data replicates it to all its neighbors and so on. With this approach, the data is certain to be delivered with the minimum delay at its destination, as one of the copies follows the shortest route between the source and the destination. However, under limited resources, including buffer storage, this strategy is not efficient as it creates an exponential number of copies of the data and fills the buffers of the entities.

Originally proposed in the context of sensing platforms, the data MULE architecture leverages the movements of entities to collect data from sensors scattered in a sparse area and deliver it to access points [SRJB03]. To improve the system performance, the MULEs can communicate with each other to either form an ad hoc multi-hop MULE network or to compose their movements and disseminate the data when in contact. CarTel [HBZ+06] and BikeNet [EML+09] have both implemented the data MULE architecture to generate sensing data using sensors mounted on vehicles and bikes equipped with sensors, respectively. Note that, in both works, the dissemination protocol was not implemented and was instead replaced by a direct delivery approach.

The Infostation model presents a data MULE application where mobile entities connect to Infostations that act as access points distributed over a geographical area [GBMY97]. When in the vicinity of an Infostation, the entities transmit at very high rates, thus trading connectivity for capacity. With the Shared Wireless Infostation Model, Small and Haas enhanced the Infostation model with the epidemic routing in the context of whale monitoring with radio-tagged whales that continuously collect biological and environmental data [SH03]. The data collected by the whales is then transmitted when the whales surface to buoys that act as Infostations. The epidemic protocol is used to exchange monitoring data when the whales are grouped together. Since the epidemic protocol generates a large number of replicas, the data accumulates at the buffers, resulting in buffer overflows and data drops. To limit the number of copies of the same file in the network, the authors use an infectious disease model that "infects" the whale with a data according to a given probability, if not already infected. This probability is chosen such that it is equal to the probability that the data will be offloaded within a target duration since its generation.

Maintaining a local control plane. Subsequent approaches proposed a local control plane to improve the performance epidemic protocol and limit the excessive usage of the resources. The Spray-and-Wait protocol bounds the number of copies to send per data using two phases [SPR05]. In the "spray" phase, a given number of copies are transmitted from the source to other entities that act as relays. In "wait" phase, the relays with a copy of the data wait to encounter the destination of the data. The Spray-and-Focus protocol [SPR07] uses the same "spray" phase as [SPR05], followed by a "focus" phase where the copies can be forwarded to other relays to help maximize a utility function (*e.g.*, delivery delay).

Other approaches proposed a more elaborated local control plane where each entity records its encounters with the other entities. This is the case of the meets and visits protocol, where each entity learns the frequency of meetings between entities and visits to certain locations in order to estimate the likelihood of forwarding the data to other entities [BBL05]. BUBBLE is a protocol that leverages the communities formed by entities such as humans to select high centrality entities (the most popular ones) and members of the community of the destination entity as relays [HCY11]. To this end, the authors rely on distributed approach where each entity detects the communities and approximates the centrality of the other entities they encounter. Other protocols such as GeOpps [LM07] and GeoSpray [SRF14] exploit the geographical information of navigation systems of vehicles to forward the data towards its destination and minimize the estimated time of delivery of the data.

Maintaining a global control plane. A global control plane allows every entity to have the same view of the network to make better decisions when composing their trajectories. To this end, the strategies with a global control plane exchange historic encounter information. PROPHET leverages the control information of the histories of all encounters to estimate a delivery predictability of the data [LDS03]. It indicates how likely an entity is to deliver a data to a destination. The data is then forwarded to the entity with the highest estimate. RAPID is another routing protocol that treats routing as a resource allocation protocols [BLV10]. It

determines the degree of replication of a message by estimating the suitability of a contact to optimize specific metrics (*e.g.*, minimize the average delivery delay) according to a replication utility. To this end, the entities exchange network state information (*e.g.*, expected meeting times with entities and past encounters) and acknowledgments among the entities. MaxProp is a routing protocol for delay tolerant networks designed in the context of the UMass DieselNet testbed [BGJL06]. DieselNet is a vehicular network that consists of 40 buses equipped with WiFi capabilities serving the surrounding area of UMass Amherst campus. With MaxProp, the data forwarding results of local decisions made by each bus according to a delivery likelihood estimation based on the history information about the past meetings with other buses. While this strategy allows having a global shared control plane, it requires exchanging large amounts of control information for large networks. Additionally, the shared view of the network is not consistent, as the control information takes time to be propagated in the network.

Discarding the remaining copies of the data. One of the problems with the previous approaches is to discard the remaining copies of the data once successfully delivered. Some approaches flood delete-lists to discard the remaining copies of the delivered data. This is the case of ZebraNet, a platform to monitor zebra wildlife in Kenya and track their location history logged in collars [JOW+02]. The data logged in the collars is reported when the zebras come in the range of base stations, which are either fixed or mobile, handheld by the researchers when they occasionally drive-by or fly-by to collect the data from the animals. The authors used the epidemic protocol to transmit the data from one zebra to another and increase the chances the data will be delivered to a base station. The delete-lists are updated with a gossip protocol whenever two zebras are in contact. Similarly, MaxProp [BGJL06] and RAPID [BLV10] both propagate acknowledgments in the network to remove stale data from the buffers and free some space to avoid buffer overflows.

The addition of a control plane helps to increase the performance of the data transfer resulting from the composition of the trajectories of the entities. The control plane gives information on how likely an entity is likely to bring the data closer to its destination. However, in real life, the composition can happen anywhere, but only the contacts that happen in a few specific areas are worth considering to improve the performance of the transfers [KWSB04, SDPG06]. The strategies we review in the following restrict the compositions of the trajectories to pre-defined locations.

2.2.1.2 Composition at pre-defined locations

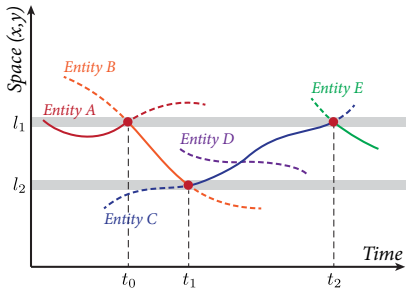


Figure 2.9. Indirect synchronous composition between two entities at pre-defined locations.

Instead of composing the trajectories of the entities anywhere in the geographic area, some strategies only allow the composition at pre-defined locations, as shown in Figure 2.9. In this case, the composition of trajectories can only happen at locations l_1 and l_2. These strategies exploit the high density of entities and their movement patterns in these areas where the contacts between the entities are more likely to happen (*e.g.*, road intersections). These areas have been exploited by context- and location-aware services such as digital graffiti [CCD+04] and floating content [OHL+11] to anchor content at specific locations.

In the case of data transfer, the use of these concentrated areas enables a finer control of the routing strategies to limit the number of replications needed when forwarding a data from an area to another. Most of the approaches we review in this section leverage a logical representation of the system with an overlay graph to route the data in the areas on the shortest path. The nodes of the overlay correspond to the areas with high densities of entities and the links represent the movements of the entities between the nodes.

Determining the locations where to compose the entity trajectories. An underlying problem related to this approach is to determine the locations with high densities of entities. In an urban vehicular setting, they correspond to intersections at junctions and traffic lights. In the context of VANETs, LOUVRE [LLHG08] and GyTAR [JSRGD09] build an overlay network on top of intersections ("landmarks" in the case of LOUVRE) where vehicles are connected to each other. When the vehicles come to the intersections, they forward the data over overlay links representing the VANET multi-hop path between two adjacent intersections. Similarly, Tan *et al.* uses the movements of vehicles traveling a transportation network to create a "vehicular backbone network" [TJWY14]. The vehicles perform "wireless switching" when they are in contact at intersections or on dual-way roads when vehicles travel in opposite directions. Sarafijanovic-Djukic *et al.* characterize the locations with high entity densities as "concentration points" (CP) [SDPG06]. Using real-life traces, the authors defined CPs as areas with at least 5% of the vehicles per day. In the context of multiple message ferries following different routes and creating a communication medium among static nodes, Zhao *et al.* propose to pass the data between the ferries at intersecting points of their routes [ZAZ05]. In this case, the ferry routes must be synchronized for the ferries to come in contact regularly. Keränen and Ott propose to transport data between airports using the smartphones of the airline passengers [KO09]. The authors rely on the scheduled flight connections at airports to compose the trajectories of the passengers.

Maintaining a global view of the network state. The entities must have a consistent and up-to-date view of the overlay network to know where and how to forward the data. There are two approaches to knowing this information. The first approach consists in assuming a low-bandwidth control channel to exchange the control messages that have low overhead. This is

the case of Tan *et al.* that uses this control channel to have the latest estimation of the overlay link metrics [TJWY14]. In the ferry work, Zhao *et al.* use this control channel to compute the intersecting routes of the ferries in an offline manner [ZAZ05]. The second approach leverages an in-band distributed approach. In [SDPG06], the entities distribute the information about the overlay using a collaborative graph discovery protocol. While this method does not provide an up-to-date information, it avoids relying on signals from the environment. LOUVRE and GyTAR rely on a peer-to-peer density discovery to populate link state table of the overlay nodes with the density information. However, in the case of a large-scale network such as the airline network proposed by Keränen and Ott, the entities cannot maintain information about all the links in the overlay. The authors argue that epidemic protocols that replicate the data are more suited for this case to forward the data [KO09].

Forwarding data when in the specific areas. With the information about the overlay, the entities can make forwarding decisions in the pre-determined areas to route the data on the shortest path to its destination. With no information on the future trajectories of the entities, [SDPG06] propose to replicate the data to multiple entities to increase the chances that at least one will arrive at the next overlay node. Once the data has reached the next entity, an acknowledgment is broadcasted to the previous entity to discard the remaining copies of the data. Tan *et al.* forward the data to minimize the delivery delay of the data and guarantee the fairness among the flows [TJWY14]. When two vehicles encounter, they choose the data flow and overlay link that are the least used of those available and the pre-allocated ones. In LOUVRE and GyTAR, the routing protocol uses the well-known Dijkstra algorithm to use the link state tables and forward the data on the overlay links with the highest density of entities [LLHG08, JSRGD09].

Using the spatial distribution of the entities to restrict the composition of the trajectories to areas where entities concentrate allows better control of the forwarding. The logical representation of the network makes the allocation of the data transfers possible and improves the performance of the resulting data transfer compared to the floating case. In the following, we review the strategies that leverage a dedicated infrastructure that temporarily buffer the data to asynchronously compose the trajectories of the entities. As a result, contrary to the synchronous case, they do not have to encounter at the same time and place to pass the data.

2.2.2. Asynchronous composition of the trajectories

While restricting the compositions of the entity movements to specific locations can improve the forwarding decisions, the movements of the entities between the locations can be unpredictable. To overcome the randomness, other approaches propose to rely on a dedicated infrastructure with intermediate nodes, either mobile or stationary, that temporarily buffer the data carried by the entities before passing it to subsequent entities. In this case, direct contacts between the entities are not required, allowing the data to be passed asynchronously.

The use of mobile intermediate nodes, such as ferries or robots, allows remote entities to communicate together, even if their trajectories do not intersect. These nodes follow pre-scheduled routes to bridge the trajectories of distant entities. In this case, the composition of the distant entities is asynchronous and floating.

Stationary intermediate nodes, such as offloading spots or throwboxes, enable the communication between two entities whose trajectories do not necessarily intersect at the same time. As a result, these nodes are pre-positioned at specific locations frequently visited by the entities. In this case, the composition of the entities is asynchronous and happens at pre-defined locations.

In the following, we review the strategies that rely on intermediate nodes to support the data transport. In the first part, we review the strategies with mobile nodes and those with stationary nodes in the second part.

2.2.2.1 Floating composition

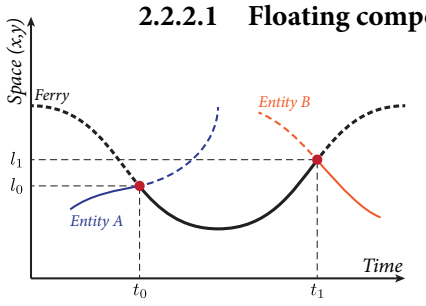


Figure 2.10. Indirect asynchronous floating composition between two entities using a *ferry*.

Mobile intermediate nodes, such as ferries or robots enable the composition of the trajectories of remote entities together. As a result, the data is passed asynchronously between the entities. In Figure 2.10, we represent the asynchronous composition of the trajectories of entities A and B using a ferry that encounters the entities. These intermediate nodes enhance the capacity of the mobile networks by providing more opportunities to compose the movements of the mobile entities. As a result, they provide an on-demand data transport service to serve the requests to transfer data of the entities.

Determining the next entity to visit. With mobile entities, the ferries must dynamically determine the next entity to serve to optimize given metrics (*e.g.*, bandwidth requirements of the entities). In the Ferry-Initiated Message Ferry (FIMF) proposed by Zhao *et al.*, the ferry follows a default route and receives requests from the mobile entities when they want to send data [ZAZ04]. The ferry then detours from its route to meet the entity and exchange data. The authors focus on the scheduling of the requests received by the ferry and propose a different heuristic to select the next entity to visit. The first heuristic always selects the nearest entity after one has been served, while the second one performs slightly better by using local optimization techniques to solve the TSP introduced in [ZA03] and recompute the route of the ferry by minimizing the expected data drops. With the Node-Initiated Message Ferry (NIMF) approach, the entities move proactively close to the ferry, which follows a pre-determined route [ZAZ04]. When an entity detours to meet the ferry, it degrades the performance of its assigned tasks. The authors propose to use a threshold based on the current duration the entities have been working on their tasks for entities to decide whether to come close to the ferry. This decision minimizes the message drop (caused by long delivery delay or buffer overflow) while limiting the detour made by the entities.

Controlling multiple ferries. The two approaches we have reviewed so far have considered a single ferry to serve the entities. Having multiple ferries creates additional challenges to scheduling the entities to visit next. This problem can be generally reduced to the dial-a-ride problem [Sav85]. Burns *et al.* borrow two multi-objective techniques from robotics for the ferries to optimize a given set of performance metrics [BBL05, BBL08, BBL06]. They consider

the subsumption composition, which prioritizes the ordered metrics and optimizes them successively, starting from the higher ones, and each up to a given performance threshold. They also consider the nullspace composition, which outperforms the subsumption composition and orders the metrics such that the optimization of these lower metrics does not affect the performance of the higher metrics.

Maintaining a global view of the network state. To decide which entity to visit next, the ferries must have an up-to-date view of the state of the network. Therefore, communications between the entities and the ferries are important to announce the positions of the ferries and the entities. In both the FIMF and NIMF, the authors propose to use long-range channels to periodically broadcast the position of the ferry and the entities. With multiple ferries, Burns *et al.* rely on a distributed approach to maintaining an approximate global view of the state of the network at each ferry. This information is exchanged and updated among the ferries when they encounter, which allows them to take online and local decisions to improve the performance of the network given a set of objectives. In both this work and the FIMF, the location of the entities is known to the ferries through either the global state information disseminated in the network or the long-range channel. However, the ferries do not predict the future positions of the entities; they only know their last positions advertised, which can lead to errors when localizing the entities.

Non-controllable dedicated entities. While the ferries or robots are dynamically controlled to decide which entity to visit, other approaches propose strategies to use the already-existing movements of dedicated entities that act as ferries to transport the data. In the context of Micro Aerial Vehicles (MAVs) used for Search and Rescue missions, Asadpour *et al.* designed a forwarding algorithm to route the data recorded by hovering MAVs with onboard cameras to a central ground station [AHGD16, AGH$^+$13, AVdBG$^+$14]. The authors introduce relay MAVs that physically carry the data between the ground station and the hovering MAVs. The algorithm relies on an estimate of the future proximity of the MAVs to the destination, calculated using a linear mobility model of the MAVs that predicts the future positions in the near future. The algorithm chooses whether to the forward data along the multi-hop shortest path to the destination if there is such a path or if the neighbor is spatially closer to the destination. Otherwise, the algorithm decides to keep the data and carry it on the MAVs until it encounters the destination or a more suitable MAV (*e.g.*, a ferry MAV).

Using mobile intermediate nodes brings more opportunities to compose the trajectories of the entities. They also enable communication between distant entities whose trajectories never intersect. As a result, their use improves the performance of the data transfers, in terms of delivery rate and delay. While these nodes bridge the connectivity between distant entities, they incur additional delays when carrying the data. The delayed deliveries are mainly due to the dynamic schedule of their route. In the following, we review the approaches that use stationary nodes to compose the trajectories of entities.

2.2.2.2 Composition at pre-defined locations

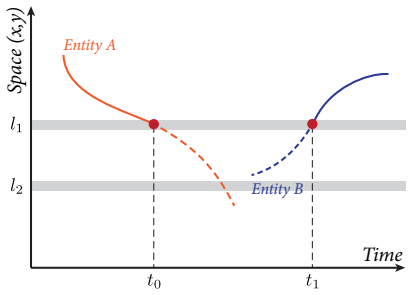


Figure 2.11. Indirect asynchronous composition between two entities at pre-defined locations using a *stationary nodes*.

In the previous section, we reviewed the approaches that rely on dedicated mobile relays to compose the remote movements of entities and increase the performance of the network. Conversely, other approaches leverage a collection of stationary nodes placed at strategic locations to asynchronously compose the movements of entities with intersecting trajectories. In Figure 2.11, we represent the asynchronous composition with stationary nodes at location l_1 and l_2 that compose the trajectories of entities A and B.

Stationary node placement. The main problem with this approach is to place the stationary nodes to efficiently compose the trajectories of the entities and optimize performance metrics such as the delivery rate or delay. The stationary nodes must be along the trajectories of the entities to efficiently compose their trajectories. Zhao *et al.* introduced *throwboxes* or small inexpensive battery-powered devices equipped with storage and wireless interfaces to enhance the delivery likelihood of mobile ad hoc networks by increasing the contact opportunities [ZCA+06]. The authors use linear programming models and heuristics to deploy the throwboxes based on the different knowledge they consider of the traffic matrix and the future contacts between the entities and the potential candidate locations of the throwboxes.

In [Cha06], Chawathe proposes to use dead drops[6], an art project that uses USB sticks placed at different locations in cities (*e.g.*, on the walls of buildings), to disseminate data using the opportunistic movements of vehicles to transport data from one dead drop to the other. The author proposes a placement of the dead drops using a greedy heuristic of the minimum-weight k-set cover problem such that it provides the required connectivity between flows of vehicles traveling different routes of interest at minimum deployment cost.

Forwarding the data to the entities. Once the stationary nodes placed at strategic locations, the next challenge is to forward the data to the entities. With throwboxes, the authors consider different routing strategies [ZCA+06]. With no assumptions on the future contacts among the entities and the throwboxes, the authors propose to use the epidemic protocol between mobile entities and throwboxes. Ibrahim *et al.* found that allowing any forwarding interactions among the mobile entities and the intermediate nodes reduces the delivery delay of the data at the expense of additional replication [INC09]. However, restricting the forwarding to the sole interactions between the mobile entities and the intermediate nodes yields poor performance in terms of delivery delay and the number of copies generated, as the source and destination must come close to an intermediate node to either transmit or receive the data. Banerjee *et al.* compare the performance of forwarding strategies with two-hop (no mobile-to-mobile communications), random Epidemic (to limit the number of copies generated by message) and

[6] `http://deaddrops.com/`

RAPID [BLV10] in the presence of different kinds of stationary intermediate nodes. The authors found that deploying x base stations (connected together by an infrastructure-based network) allows reducing the average packet delay by a factor of two. The same reduction requires $2x$ mesh nodes (connected together by long range wireless links) and $5x$ relays (acting as throwboxes [ZCA+06]).

The forwarding problem was extensively studied in KioskNet a low-cost alternative that leverages the movements of vehicles between rural kiosks deployed in remote regions with limited Internet connectivity and offers services such as email and land records [SKZ+06, GFO+07, GDF+11]. This creates a mechanical backhaul that transports data between an Internet gateway located in a nearby city and the kiosks. The kiosks serve as routers to forward the data to other kiosks, thus creating an overlay network with the kiosks as intermediate nodes and the movements of the vehicles as links. The authors propose to use different routing strategies, including flooding (does not scale), reverse-path forwarding (fragile in case of failures), and link-state routing, which uses the expected delay between two kiosks as the link weight.

Following the same architecture as KioskNet, Demmer and Fall propose DTLSR, which modifies the Link State Announcements (LSAs) with long lifetimes to take the changes in the connectivity into account [DF07]. This avoids considering the node as unreachable if the link to this node is unavailable at the lime of the LSA is computed, as it is the case in legacy link-state routing. The data is then routed using a shortest-path algorithm (*e.g.*, Dijkstra) such that the estimated expected delay is minimized.

In TrainNet, the authors also formulate the routing problem of carrying data on trains as a fair resource allocation problem [ZAC10]. Since the trains have limited storage available and only a limited amount of data can be exchanged at stations, the routing problem maximizes the throughput and hard drive utilization, while minimizing the delay and data losses.

Energy-efficient forwarding. Banerjee *et al.* propose a framework to enable power savings at throwboxes equipped with solar panels [BCL07]. In particular, they focus on improving the neighbor discovery phase, which is the primary source of energy consumption, by using two radio channels to separate the neighbor discovery task from the data transfer task. A long-range channel is in charge of the neighbor discovery task and efficiently listens for contacts (duty-cycle), while a high-bandwidth radio remains asleep. On a contact with the long-range channel, the throwbox determines whether the contact is fruitful by (*i*) predicting the future mobility of the entity and (*ii*) determining whether to accommodate this contact or the future ones based on the current level of energy available and the estimated duration of this contact.

Combination with delivery services. The same approach is also used in combination with delivery services, which we introduced in Section 2.1.3. Wang *et al.* propose a centralized architecture with either a central server or geo-distributed servers [WSG+04]. The servers aggregate the data received from parcels and dispatches it to the final users or route it across the servers, still using the postal services. Adding multiple servers allows the users to send the data to the closest server, thus reducing the costs of sending the data. Cho and Gupta propose

a joint use of the Internet with postal services to minimize the cost and latency of the transfer [CG10, CG11]. To this end, they leverage an overlay of the sites connected by both shipping links and Internet links. The overlay links are represented by a capacity, cost and transit time, which vary over time in the case of shipping links (*e.g.*, both the cost and transit time vary as a function of the level of service — Overnight, Two-day, Ground). The authors reduce their problem to a flow over time problem [FS07], which they manage to make tractable with optimizations for large-scale networks. In every case, their system outperforms the single use of the Internet or the shipping method to transfer data.

Message ferry case. Finally, in the case of multiple ferries, the authors propose to use a dedicated infrastructure to pass the data between ferries following different routes [ZAZ05]. The stationary nodes are then placed at the intersections of the ferry routes to compose the trajectories of the ferries. Bin *et al.* design a ferry route such that a single ferry visits the mobile entities without any online collaboration at determined locations for determined durations with a certain minimum target probability [BTAZ06]. While this approach does not leverage stationary nodes, it results in an asynchronous forwarding restricted to specific locations. To design the route, the algorithm first determines the way-points among a set of candidate points and the corresponding waiting times at these way-points such that the ferry meets the other nodes with a given probability (either while moving or while waiting at a way-point). Second, the algorithms determine the route to follow such that it minimizes the weighted sum of the waiting times and the total distance of selected way-points from the center of the region.

Stationary intermediate nodes have been used in wide-range of applications and give the potential to increase the capacity of the resulting data transfers. Their placement is important to capture the trajectories of the entities to have opportunities to compose them together. This is the approach we have considered with the offloading spots to realize the offloading system we present in this thesis.

2.3 | *Relevance to the thesis*

In this section, we discuss the relevance of each part of our work with respect to the state-of-the-art we presented in previous sections.

Vehicular data offloading. In the first part of our work, we rely on offloading spots that act as intermediate relay nodes similar to those used to asynchronously compose entity trajectories at pre-defined locations such as throwboxes [ZCA+06]. In this case, these intermediary nodes create additional contact opportunities and bring the data closer to its destination. They also introduce non-randomness in an environment where entity trajectories are difficult to predict. The authors showed that these nodes are an effective way to compose the entity trajectories and improve the overall throughput and delivery delay.

Our system aims to transfer large amounts of delay-tolerant data, between remote locations. Examples of large data transfers over the Internet range from exchanges of large scientific

datasets to distribution of high-resolution movies. Some protocols have been developed to handle bulk data transfers of several Terabytes per day, such as GridFTP [ABB+03] or NetStitcher [LSYR11]. The latter takes into account unused bandwidth in inter-data center networks (during low link utilization periods) by using multi-path and store-and-forward scheduling at intermediate nodes for bulk transfers.

While these previous approaches allow transferring bulk data, we propose to *physically offload* data by creating an alternative communication channel consisting of existing vehicle movements. Several works proposed to create such a channel by relying on the movements of entities such as airline passengers [KO09], trains [ZAC10], or even postal services [WSG+04, CG11, LSS+13]. While these approaches are comparable with our work in terms of scale, throughput, and delay, they are difficult to compare under equivalent settings, including the number of entities involved and trips made, as well as their coverage.

Vehicular cloud services. In the case of the first extension, we rely on the schedule movements of public buses to distribute user files among repositories. This work is closely related to KioskNet [SKZ+06] and DakNet [PFH04], although they do not provide any performance evaluation and they take place in rural environments with fewer bus routes and trips available. In our system, the bus act as message ferries that connect the repositories together [ZAZ04]. While they are not controllable, they have the same purpose to connect remote nodes together.

The consumer/producer behavior of the mobile users is similar to the TierStore system, which acts as a Network File System (NFS) for applications with limited bandwidth or challenged network environments [DDB08]. User files are stored in persistent storage repositories that lie at the core of the system (*i.e.*, static nodes similar to our repositories). Updates are applied locally at the persistent repository and distributed to other nodes. The system also uses the movements of mobile nodes to distribute updates to and from the nodes using static multicast. On the other hand, Ott and Pitkänen proposed to use the mobile nodes to temporarily store content by means of caching [PO07, OP07] to decrease the latency when retrieving a content. This approach is similar to the one proposed by Haggle, a *Pocket Switching Networking* (PSN) system that enables caching of content at intermediate nodes to also reduce the delivery delay [SCHD06].

The placement of the repositories in this extension is similar to the placement of throwboxes, where the authors select the optimal locations for throwboxes from a set of candidate locations (*i.e.*, centers of square cells dividing the geographical space) to maximize the total data rate, given different knowledge on contact opportunities and traffic load for various routing strategies (*i.e.*, single-path, multi-path, and epidemic) [ZCA+06]. However, our placement does not account for the routing strategy, which makes it more related to the one proposed by Chawathe in the context of dead drops to disseminate data [Cha06]. In this case, the drop boxes have the same behavior as our repositories and data is disseminated using the (known) movements of vehicles. The placement of dead drops must provide the required connectivity among them at minimum deployment cost and is solved using a heuristic of the minimum-weight k-set cover problem fitting the model.

Finally, in the second extension, we leverage pre-defined locations selected such that vehicles encounter for a duration long enough to transfer large amounts of data through vehicle-to-vehicle communications. While these locations are similar to those used for wireless switching by Tan *et al.* to create a "vehicular backbone network" [TJWY14], or those used by Sarafijanovic-Djukic *et al.* to perform data forwarding [SDPG06], our work studies the capacity of the contacts at specific locations.

Logical view of the vehicle movements. Throughout the thesis, we analyze and characterize the vehicles' movements between locations (for the data offloading part and the first extension) or the locations where vehicles are in contact with one another (for the second extension) with a logical view representing the dynamics of the network. Our contribution lies in the fact that we use this view to translate movements into networking quantities and enable the allocation of data transfers. This is the case of the offloading overlay that mitigates the complexity of the road network and represents the flows of vehicles traveling between the offloading spots as resources to allocate. We leverage this logical view in the extensions to deploy the repositories (in the case of the first extension) and the contact hotspots (in the case of the second extension).

Centralized architecture. Instead of using a distributed control plane as is the case with most of the related approaches, we leverage a centralized architecture with a controller that has a holistic view of the system to efficiently allocate vehicular resources. In the case of the data offloading, the controller then derives forwarding states from the allocation output and installs them at the offloading spots. With these states installed, the offloading spots are able to decide which buffered data to load on the stopping vehicles. This controller performs additional management tasks, including reliable data transfers, file distribution control (in the case of the first extension), and virtual machine migrations control (in the case of the second extension).

3 Assessing the concept of vehicular data offloading

The objective of this chapter is to assess the concept of data offloading. The main step toward the feasibility of this concept can be stated as follows:

> *Given the demands to offload data transfers from a conventional data network, how does one select the road network paths and the flows of vehicles matching the data transfer requirements to maximize the revenues of offloading traffic on vehicles over the use of a conventional data network?*

We first give an overview of the concepts underlying vehicular data offloading and the operations of the offloading infrastructure. We then present a reference scenario in the context of a network of charging stations for Electric Vehicles (EVs). Data is unloaded from or loaded on the EVs, without the drivers being aware, while charging their batteries as they usually do. The scenario fits with the recent emerging trends in the EV market, as car manufacturers and charging stations operators are seeking additional revenues beyond sales from added services offered while EVs are being charged [Lim14].

We propose an allocation procedure that maximizes the cost-benefit of offloading traffic on the road network compared to transferring the same amount of data on a conventional data network. We formulate this allocation procedure as a *revenue maximization problem*. To solve this problem, we use a linear programming model that selects the flows of vehicles that maximize the revenue resulting from offloading data over the road network.

To solve the revenue maximization problem in a reasonable computational time, we propose a mapping algorithm that creates a logical representation to mitigate the complexity of the road network. In this representation, nodes corresponding to the offloading spots are connected by logical links. A logical link corresponds to the flows of vehicles traveling the road segments connecting two adjacent offloading spots in the road network. We propose a method that derives from the road traffic counts a characterization of the logical links in terms of delay, capacity, and data loss. The offloading overlay mitigates the complexity of the substrate network and makes the allocation of vehicles tractable.

We finally provide numerical results using actual road traffic counts in the road network of France.

As a summary, the main contributions of this chapter are:

- **Vehicular offloading** (Section 3.1). We describe the principle of offloading traffic on vehicles. We introduce the concept of *offloading spots*, which refer to storage facilities

deployed along the roads. We present a reference scenario involving offloading data on Electric Vehicles while charging their batteries at stations acting as offloading spots.

- **Road map space reduction** (Section 3.2). We design a mapping algorithm to mitigate the complexity of the road network. The output of our algorithm is an *offloading overlay* that gives a comprehensive representation of the underlying resources.
- **Revenue maximization problem** (Section 3.3). We formulate the *vehicle flow allocation problem* as a novel Linear Programing (LP) model. We solve the LP model so as to determine the logical path that maximizes the expected revenues from offloading data on the road network.
- **Real-world assessment** (Section 3.5). We evaluate our revenue maximization model on the French road network using *actual road traffic counts*. The results show that the existing mobility of vehicles can accommodate transfers in the order of one Petabyte during a one-week time window.

The remainder of the chapter is structured as follows. In Section 3.1, we present the concept of vehicular offloading by providing an overview of the operations to offload data over the road network. We also present a reference scenario involving electric vehicles and battery charging stations. In Section 3.2, we describe the map reduction procedure that mitigates the complexity of the road network. We introduce the revenue maximization problem in Section 3.3. We then formulate the revenue maximization problem as a linear programming model in Section 3.4. In Section 3.5, we assess the concept of vehicular offloading by using actual road traffic counts for the French road network. Finally, Section 3.7 draws the conclusions of this chapter.

3.1 | *Vehicular data offloading*

3.1.1. Offloading infrastructure operations

In the section, we give a schematic overview of the operations of the infrastructure in charge of offloading large amounts of delay-tolerant data over the road network. We target data transfers lasting several days that result from provisioning or maintenance activities required for virtual machine migrations or offline backup between data centers. Figure 3.1 depicts the operations on a fragment of the road network.

Private vehicles are equipped with one or more removable storage devices such as magnetic disks or other non-volatile solid-state storage devices. We assume that the content of the storage devices is not accessible by the drivers and the data encrypted when piggybacked onto the vehicle. We provide in Section 3.6 ways to guarantee secure transportation of the data. Vehicles also embed one or more communication network interfaces.

The flow of vehicles so equipped acts as a mechanical backhaul connecting a collection of offloading spots, as depicted in Figure 3.1. The term *offloading spot* refers to a fixed device offering short- to medium-term data storage, located where vehicles stop for long periods of time as part of their travel route. Examples of such locations include on-street parking spots, parking lots,

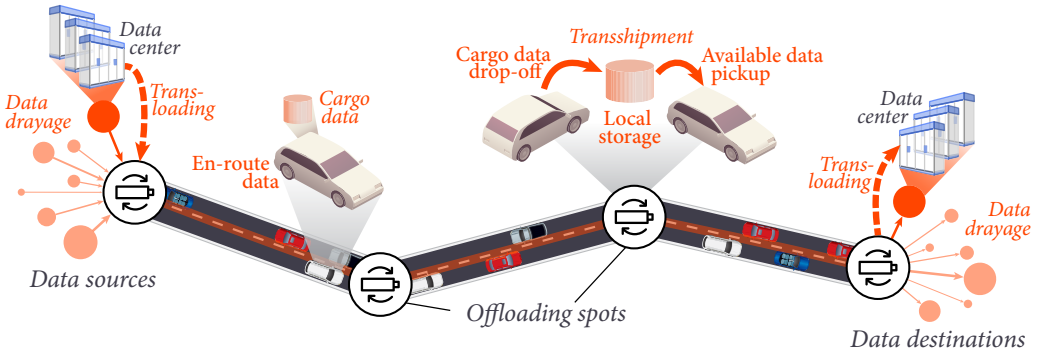


Figure 3.1. Overview of the operations to offload large amounts of data over the road network.

or gas stations. In the case of electric vehicles, the offloading spots may be located at the battery charging stations such as the ones operated worldwide by ChargePoint[1]. Offloading spots are equipped with wireless communication interfaces to support short-range radio data exchanges with vehicles while stopped.

Offloading spots act as data relay exchange points where data is *transloaded* from a conventional data network to the closest offloading spot using a *drayage* system and stored until shipped to its destination by vehicles. Different techniques to implement a data drayage system are presented in Section 3.6. We take advantage of the parking time of the vehicle to opportunistically load data on or off their storage devices.

Vehicles unload their data cargo while they are stopping at an offloading spot if they are heading in a different direction than the destination of the data they carry. The data is stored until transferred to a subsequent empty vehicle heading towards the intended destination. As a result, the data may be transferred on multiple vehicles following different trajectories before reaching its destination. Splitting the data path across different vehicles makes the transfers less vulnerable to hijacking attempts on vehicles. To mitigate the effect of vehicles changing their travel directions or having accidents, offloading spots load the same data on multiple vehicles, either at the first offloading spot or at each intermediate offloading spot.

The data is therefore "hitchhiked" hop-by-hop through the network of offloading spots before reaching its final destination. Transfers of offloaded data result from the vehicles' routine journeys interspersed with stops at the offloading spots as part of their line of travel. Offloading spots thus remove the need of relying on the only vehicles making the trip all the way from the source to the destination of a data transfer.

The data offloaded from conventional networks follows the same path as the flow of vehicles traveling the road segments connecting sequences of offloading spots.

For each offloading demand, the road network path is determined by solving the vehicle flow allocation problem we formulate as a multi-commodity flow allocation model in Section 3.3.

[1] `http://www.chargepoint.com/`

3.1.2. Business model, assumptions, and roles

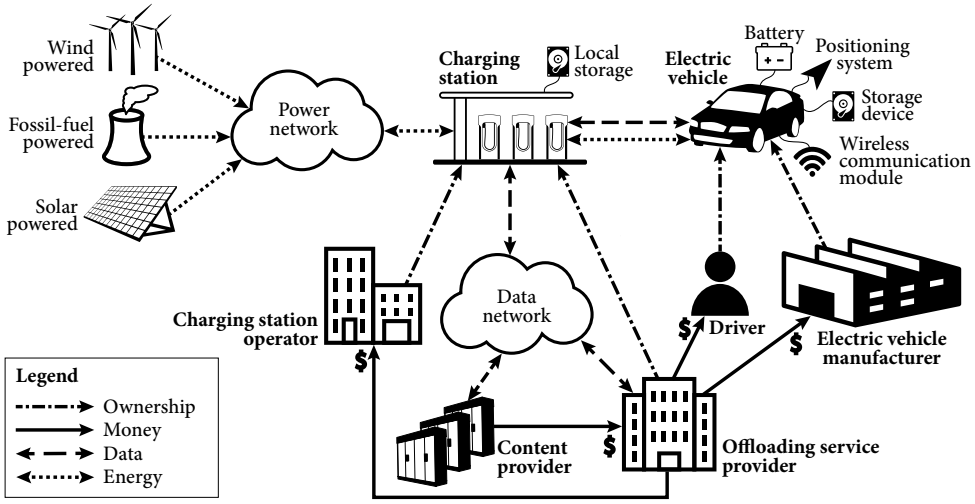


Figure 3.2. Business model for the vehicular data offloading concept.

In this section, we present the business model we will use throughout this thesis. We represent the different actors involved in Figure 3.2, including (*i*) Electric Vehicles (EVs) and their owners, (*ii*) the offloading service provider (that can be the EV manufacturers themselves), (*iii*) the charging station operator, and (*iv*) a content provider that operates multiple data centers.

This model provides support for the modeling and analysis of our proposal in practical settings, including the driving range of EVs (*e.g.*, 320 km for the Tesla Model S) and the charging time of the batteries (*e.g.*, 20 minutes when charging at a Tesla supercharger[2]).

The electric vehicles are equipped with one or more data storage devices, such as hard drives or other non-volatile solid-state devices. The term "electric vehicle" EV refers to private vehicles propelled by one or more electric motors powered by a rechargeable battery. The charging stations act as *offloading spots* where the data is seamlessly loaded on or unloaded from the onboard storage of the vehicles using short-range radio while they charge their battery.

The offloading service provider, if different from the EV manufacturers, proposes a "get paid to drive" program to the car owners. The service provider installs the storage devices and the vehicle owners receive a monthly fee or a discount on the cost of charging their vehicle in exchange for driving their normal routine. The discount rate is negotiated with the charging station operator (*e.g.*, ChargePoint who operates a worldwide network of EV charging stations offering cloud-network services[3]) and is calculated based on the driving pattern including coverage and mileage. If the EV manufacturers take on the role of service provider, vehicles are equipped as standard with onboard storage and the service is provided without involving nor compensating the vehicles' drivers. The service provider charges the content provider for the amount of data to offload on the road network and shares the revenues with the charging station operator.

[2] `https://www.tesla.com/supercharger`

[3] `http://www.chargepoint.com/`

The offloading spots feature storage capabilities where data is transloaded from the conventional data network and temporarily stored until transferred to a charging vehicle. The service provider monitors the status of the offloading spots, which includes the amount of available storage and the destination of the data cargo in transshipment. To improve the forwarding decisions, the service provider also queries the stopping vehicles' positioning system to determine their current geographic destination. The historical locations are stored in a geographic location database managed by the service provider to help the offloading spots predict the remaining itinerary of the passing vehicles. In Section 3.6, we present a list of probabilistic tools for inferring the remaining route of a vehicle knowing its navigation history and discuss the concerns regarding the drivers' and passengers' privacy.

Upon receiving a demand to offload a data transfer from a content provider, the service provider computes the road network paths that can accommodate the data transfer requirements and how much data to allocate to each flow of vehicles. The road network path consists of a sequence of offloading spots, each configured with the list of actions to perform on each vehicle traveling in the direction of the next-hop offloading spot. Each action defines the behavior that the offloading spot operator needs to perform with the data belonging to the offloaded transfer. The corresponding data can either be already stored at the offloading spot or carried by the charging vehicle. Common actions include loading data cargo on or off the vehicles while charging their battery. The service provider defines these actions based on the information the offloading spot operator reports on the flows of vehicles passing through the offloading spots. For each vehicle stopping by an offloading spot, the offloading spot operator matches the destination of the vehicle against the data transfers currently offloaded on the road network and performs the actions as dictated by the service provider.

3.2 | Road map reduction

Recall that our objective in this chapter is to assess the vehicular offloading concept by comparing the cost of using the offloading infrastructure including the offloading spots compared to the cost charged by Internet providers for transferring the same amount of data. We need to determine the routes followed by the vehicles carrying the data and offering the same performance as the one resulting of transferring the same amount of data on the Internet.

To determine those routes, we present a revenue maximization model in Section 3.3. Before solving this model, we need to cope with the scale of the road network. The high degree of complexity of the road network's topology and the large number of vehicular trips make the revenue maximization problem computationally intractable.

In this section, we present a mapping algorithm that reduces the complexity of the road network. The output of our algorithm is an offloading overlay, which refers to a logical representation that captures the characteristics of the road network. We provide a performance evaluation that measures the reduction factor resulting from the mapping algorithm by comparing the complexity of the offloading overlay over the actual road network.

3.2.1. Road network

We represent the road network by a directed graph $G^R = (N^R, L^R)$, where N^R and L^R denote the set of physical nodes and links, respectively. The set of nodes $N^R = N^J \cup N^S$ consists of two subsets: the set of road junctions (N^J) and the set of charging stations (N^S). A road junction refers to a location where vehicles can change their direction of travel. A link in the road network corresponds to a road segment connecting two adjacent junctions or a junction and a charging station. We consider the road segments and the traffic flowing in both directions homogeneous as they share the same profile in terms of capacity. For a road segment $(a, b) \in L^R$, let v_{ab} be its nominal volume of vehicles (vehicles per unit of time), c_{ab} be its capacity (vehicles per unit of time), and $t_{ab}(v_{ab})$ be its corresponding travel time that depends on the nominal volume of vehicles.

We use publicly available datasets to characterize the road network. These datasets feature traffic volumes on the road segments, expressed in terms of Annual Average Daily Traffic (AADT). AADT is the total volume of traffic traveling on a road segment in both directions for one year, divided by the number of days in the year. As a result, the capacity c_{ab} of a road segment $(a, b) \in L^R$ are fixed to the AADT value of the dataset for this road segment.

3.2.2. Offloading overlay

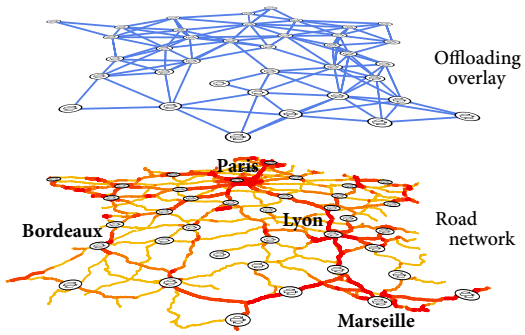


Figure 3.3. The offloading overlay resulting from the deployment plan of battery charging stations acting as offloading spots for the French road network (the width and darkness of the road segments denote the vehicle density).

The offloading overlay provides a logical representation of the road network. More specifically, the overlay mitigates the combinatorial explosion when enumerating the simple road paths in the road network, which becomes quickly intractable when the length of the paths increases. By providing a logical representation of the vehicle flows on the road paths connecting the offloading spots, the offloading overlay reduces the number of paths to enumerate.

We represent the offloading overlay by a directed graph $G^O = (N^O, L^O)$, where N^O and L^O denote the set of logical nodes and links, respectively. A logical node of N^O is an offloading spot. A logical link of L^O represents the road path (*i.e.*, a sequence of road segments) connecting two adjacent offloading spots in the road network. Note that, multiple road paths may connect two adjacent offloading spots, in which case, the logical link aggregates these paths. In Figure 3.3, we show an example of realization of an offloading overlay on top of the French road network.

In the following Section 3.2.3, we characterize a logical link $(i, j) \in L^O$ with network quantities, that is by the weighted travel time $t(i, j)$, aggregated capacity $c(i, j)$ and data leakage $l(i, j)$. The data leakage refers to the loss rate on logical link $(i, j) \in L^O$ and accounts for the proportion of vehicles that fail to deliver the data to the next offloading spot j because of errors

in the prediction of the direction of the vehicle or accidents. We assume that offloading spots are not constrained by the amount of transfers they can serve and have the adequate storage capacity so that the overall service is stable.

3.2.3. Characterization of the offloading overlay

In this section, we characterize the logical links of the offloading overlay into network quantities. The network quantities we consider in the following are relevant to solve the revenue maximization problem.

Travel time $t(i,\ j)$**.** The travel time tab of $(a,\ b)$ is given by the Bureau of Public Roads (BPR) function defined as [Uni64]:

$$t_{ab}(v_{ab}) = t_{ab}(0)\left[1 + \alpha\left(\frac{v_{ab}}{c_{ab}}\right)^{\beta}\right], \tag{3.1}$$

where α and β are BPR parameters that depend on the road profile ($\alpha = 0.15$ minutes and $\beta = 4.0$ are typically used) [Nat00].

We can deduce from Equation 3.1 the travel time of physical path p, denoted t_p, which is the sum of all travel times of the road segments that compose the path (we do not consider any turning delays at junctions):

$$t_p = \sum_{(a,\,b)\in p} t_{ab}(v_{ab}). \tag{3.2}$$

From Equation 3.2, we deduce an expression of the average travel time $t(i,\ j)$ experienced on the r physical paths between nodes i and j, weighted by the road traffic flow v_p on each path p:

$$t(i,\ j) = \frac{\sum_{p\in S_{ij}} t_p\, v_p}{\sum_{p\in S_{ij}} v_p}, \tag{3.3}$$

where S_{ij} is the set of all simple physical paths between i and j (*i.e.*, with no cycles in the path).

Capacity $c(i,\ j)$**.** The capacity $c(i,\ j)$ of the overlay link $(i,\ j)\ \in\ L^O$ depends on the sum of the traffic flows v_p of the simple paths between offloading spots i and j (*i.e.*, the number of vehicles per unit of time going from i to j on path p). The capacity $c(i,\ j)$ of the overlay link also depends on the market penetration ratio $\mathcal{M}$ of the vehicles participating in the data offloading and the storage size $\mathcal{S}$ on each vehicle. We assume that all vehicles are equipped with a storage device of the same size $\mathcal{S}$.

$$c(i,\ j) = \mathcal{M}\mathcal{S}\sum_{p\in\mathcal{P}^{ij}} v_p. \tag{3.4}$$

In the performance evaluation presented in Section 3.5, we derive the value of v_p for each path p from the French road network dataset. This dataset provides the real traffic counts in terms of AADTs for the road segments composing each path p.

Leakage $l(i, j)$. The leakage $l(i, j)$ (comprised between 0 and 1) of logical link $(i, j) \in L^O$ represents the proportion of data that is lost between offloading spots i and j. The leakage increases as more vehicles carrying data prematurely exit the road (*e.g.*, the vehicles may exit the highway before reaching the offloading spot or an accident may have occurred). The leakage depends on characteristics that are inherent to the physical paths mapped in the offloading overlay. Additionally, the leakage accounts for the errors inherent to the prediction of the direction of the vehicles when loading data at an offloading spot.

3.3 | *Revenue maximization model*

In this section, our objective is to assess the concept of vehicular offloading. The main step towards the feasibility of this concept can be stated as follows: Given a demand to offload a data transfer from a conventional data network, how does one select the road network paths and the flows of vehicles to match the demand requirements.

We introduce the revenue maximization model that maximizes the cost-benefit of offloading traffic on the road network instead of using a conventional data network. We assume that the data is replicated in multiple copies each transferred over different vehicles to the next offloading spots. We devise two replication models resulting in two separate formulations of the revenue maximization models.

3.3.1. Offloading demands

An offloading demand d_{st} represents a request to offload a data transfer from a source offloading spot $s \in N^O$ to a target offloading spot $t \in N^O$. The demand is characterized by the amount of data β_{st} and the deadline τ_{st} before which the transfer should be completed. Recall from Section 3.2.2, that we need to account for the vehicles that fail to reach the next offloading spot on time. Thus, each offloading demand is also characterized by a leakage tolerance l_{st}. We denote by $\mathcal{D} = \{d_{st}\}$ the matrix of the offloading demands indexed by the pairs of source and destination offloading spots (s, t) of the corresponding data transfers. The leakage tolerance $0 < l_{st} \leq 1$ refers to the data loss ratio tolerated during the transfer on the road network, provided that the data is replicated at the source according to redundancy techniques (*e.g.*, Reed-Solomon or RAID) [CLG+94], combined to the use of error correcting codes [MS77]. The leakage tolerance is consistent with the data encoding mechanism. Note that, β_{st}, the total amount of data to be transferred between s and t, includes the redundant data introduced by the redundancy techniques.

In the vehicle flow allocation model, p denotes a logical simple path (*i.e.*, without any cycles) defined as the ordered list of offloading spots connected by logical links. $\mathcal{P}_{st}$ denotes the set

of all logical simple paths from s to t. We characterize the logical paths by the travel time $t(p)$ and the data flow (throughput) $f(p)$ allocated on logical path $p \in \mathcal{P}_{st}$ for a given offloading demand d_{st} resulting from the vehicle flow allocation problem. The logical path travel time corresponds to the average time it takes to transfer one data cargo from one extremity of the path to the other.

3.3.2. Replication models

We assume that the sequence and the properties of the links comprising the path followed by a data transfer remain unchanged for the time of the observation. We consider that offloading spots undertake two different behaviors when incoming vehicles unload their data cargo:

- **Local replication (*rep*).** This model assumes that all offloading spots replicate data on multiple outgoing vehicles. For a given offloading demand d_{st} and for each adjacent offloading spot j, offloading spot i replicates data $\rho_{st}(i,\, j)$ times vehicles traveling on logical link $(i,\, j) \in L^O$ (*i.e.*, the same data is loaded onto $\rho_{st}(i,\, j)$ vehicles heading to j). This logical link is characterized by the leakage $l(i,\, j)$. The amount of replication needed is calculated such that it satisfies the leakage tolerance l_{st} of the offloading demand.
- **Source replication (*nrep*).** The second model assumes that none of the offloading spots replicate data on the outgoing vehicles, except at the source offloading spot. It takes into account the combination of the leakage on the logical links that compose logical path p, denoted by $l_p(i,\, j)$ for the resulting leakage at logical link $(i,\, j)$. Here, we assume the source can characterize the entire paths to be allocated using traffic forecasting techniques before the transfer happens [dDOW11, She85]. For path p and demand d_{st}, the source offloading spot s replicates the data $\rho_{st}(p)$ times such that it satisfies the leakage tolerance l_{st} of the offloading demand.

3.3.3. Cost model

We aim to measure the cost-benefit of offloading data over a solution relying on conventional data networks when delivering the same amount of data. We consider a offloading demand d_{st} characterized by the amount of data to transfer β_{st} and the transfer duration τ_{st} we target. Our model describes the cost-benefit of a data transfer, defined as the difference between the price charged by an Internet Service Provider (ISP) for transferring β_{st} within τ_{st}, and the operational costs induced by transferring data with the offloading infrastructure. In the following, we define the Internet-based price and the operational costs.

- **Internet-based cost.** γ_{st} is the unit cost to send a bit of data from s to t over the legacy Internet (*e.g.*, a leased line between s and t). As an example, γ_{st} may vary in function of the distance between s and t. Without loss of generality, we assume that γ_{st} is linear with the volume of data transferred. γ_{st} may also be seen as the savings one can achieve by using our offloading infrastructure instead of an Internet-based solution.
- **Operational costs.** α_i is the operational cost needed for the maintenance of the storage facility at offloading spot i, as well as the financial compensations given to the drivers of the vehicles involved in the data offloading. We assume α_i is also linear with the volume

Table 3.1: Table of notations for the revenue maximization problem.

Variable	Meaning
d_{st}	Offloading demands between source s and destination t
τ_{st}	Delay tolerance of demand d_{st}
l_{st}	Leakage tolerance of demand d_{st}
β_{st}	Amount of data transferred for demand d_{st}
$\mathcal{P}_{st}$	Set of simple paths between s and t on the offloading overlay
$\mathcal{S}$	Storage capacity of the vehicles (cargo size)
$\mathcal{M}$	Market penetration ratio
γ_{st}	Internet-based cost for demand d_{st}
α_i	Operational cost at offloading spot i
δ_i	Waiting time at offloading spot i (to load data)
$f(p)$	Flow allocated on logical path p
$t(p)$	Travel time of logical path p
$\rho_{st}(p)$	Replication factor for demand d_{st} on path p
$\rho_{st}(i,\, j)$	Replication factor for demand d_{st} on link $(i,\, j) \in L^O$
$t(i,\, j)$	Weighted travel time of logical link $(i,\, j) \in L^O$
$c(i,\, j)$	Capacity of logical link $(i,\, j) \in L^O$
$l(i,\, j)$	Leakage of logical link $(i,\, j) \in L^O$
$l_p(i,\, j)$	Combined leakage of logical path p at link $(i,\, j) \in L^O$

of data exchanged at offloading spot i. The value of α_i depends on the replication model we consider. α_i^{rep} corresponds to the local replication model and α_i^{nrep} corresponds to the source replication model. The local replication model will require the data to be stored for longer periods of time. Each copy is held until it can be transferred to a vehicle matching the destination of the data. Hence, the need for extra storage facilities will induce higher operational costs for the local replication model α_i^{rep}, compared to the source one α_i^{nrep}.

Note that we do not explicitly consider the Capital EXpenditure Costs (CAPEX) to deploy the infrastructure required to offload data over the road network. The CAPEX includes the storage facilities at the offloading spots and the storage devices installed on the vehicles. Instead, we consider the infrastructure already deployed and exploited by the offloading spot operator (*e.g.*, the charging station operator). Hence, the CAPEX costs are amortized and thus accounted for in the operational costs we consider.

In the following, we present our revenue maximization model as a Linear Programing model derived from a Multi-Commodity Flow allocation model. The model maximizes the overall cost-benefit of offloading the entire set of demands $\mathcal{D}$ between each pair of (source, destination). Our cost model constraints the objective by both the delay tolerance τ_{st} and the capacity $c(i,\, j)$ of the logical links. We summarize in Table 3.1 the variables we use in our model.

In the following, we formulate the revenue maximization problem using the offloading overlay with a linear programming model that aims at maximizing the cost-benefit of offloading data over the road network. We consider a snapshot of the system, *i.e.*, a state of a system characterized by given parameters valid for the duration of the data transfers (offloading demands, road traffic flows, travel time and leakage on logical links).

3.4 | *Data offloading revenue maximization*

In Section 3.3, we described the revenue maximization model we consider that maximizes the cost-benefit of offloading data transfers on the road network. Here, we present the two revenue maximization models which differ depending on the data replication models used for the data transfers.

The travel time $t(p)$ experienced on logical path $p \in \mathcal{P}$ is the sum of the travel times $t(i,\, j)$ on the logical links of p and the waiting time δ_i at each intermediate offloading spot on the path:

$$t(p) = \sum_{(i,\, j) \in p} \big(t(i,\, j) + \delta_i\big). \tag{3.5}$$

For the same offloading demand, the transfer of an amount of data β_{st} is constrained by the delay tolerance τ_{st} and does not depend on the data replication model. Hence, the delay constraint is the same for both models. The delay to transfer β_{st} between s and t is the sum of the duration to transfer this quantity over all logical paths between s and t and the average travel time experienced on these paths, weighted by the flow on each path. The delay constraint is expressed as:

$$\frac{\sum_{p \in \mathcal{P}_{st}} f(p) t(p)}{\sum_{p \in \mathcal{P}_{st}} f(p)} + \frac{\beta_{st}}{\sum_{p \in \mathcal{P}_{st}} f(p)} \leq \tau_{st}. \tag{3.6}$$

Equation 3.6 is not a linear expression, which can lead to some difficulties to solve the linear optimization problem. But it can be rewritten as the following linear expression:

$$\sum_{p \in \mathcal{P}_{st}} f(p)\, (\tau_{st} - t(p)) \geq \beta_{st}. \tag{3.7}$$

3.4.1. The "local replication" model

Recall that, in this model, every offloading spots store and replicate data. Consider a given offloading demand d_{st} with leakage tolerance l_{st} and $p \in \mathcal{P}_{st}$ a path between s and t. If $\rho_{st}(i,\, j) f(p)$ data is transmitted on overlay link $(i,\, j) \in p$, $f(p)$ data is received at destination offloading spot j. Here, the replication factor $\rho_{st}(i,\, j)$ is calculated by each offloading spot i that sends data to remote offloading spot j on logical link $(i,\, j) \in L^O$ as a function of l_{st} and $l(i,\, j)$ as follows:

$$l(i,\, j)^{\rho_{st}(i,j)} \leq l_{st} \implies \rho_{st}(i,\, j) \geq \frac{\log(l_{st})}{\log(l(i,\, j))}. \tag{3.8}$$

The flow on logical link $(i,\, j) \in L^O$ is expressed as the sum of all flows sent by offloading spot i to offloading spot j on logical link $(i,\, j)$ for offloading demand d_{st}. The flow is constrained by the capacity of the logical link $c(i,\, j)$:

$$\sum_{s,\,t\in\mathcal{D}} \sum_{\substack{p\in\mathcal{P}_{st}\\ p\ni(i,\,j)}} \rho_{st}(i,\,j)\, f(p) \leq c(i,\,j). \tag{3.9}$$

Recall that, the cost-benefit is expressed as the difference between the cost of an Internet-based service (using the cost factor γ^{st}) and the operational costs of the offloading infrastructure (costs α_i^{rep} at each offloading spot i). Our objective is to maximize the cost-benefit of offloading data, expressed as follows:

$$\sum_{s,\,t\in\mathcal{D}} \beta_{st}\gamma_{st} - \sum_{i\in N^O} \alpha_i^{\text{rep}} \sum_{p\ni i} f(p). \tag{3.10}$$

One can note that the contribution to the value of Equation 3.10 of each $f(p)$ is negative. Besides, the only lower bonding constraint on $f(p)$ is Equation 3.7, which can be interpreted as a requirement on the total flow between each pair $(s,\,t)$. Therefore, maximizing the cost-benefit tends to minimize $\sum f(p)$, and Equation 3.7 is tight at optimality. It is thus possible to replace β_{st} in Equation 3.10 as follows.

$$\sum_{s,\,t\in\mathcal{D}} \sum_{p\in\mathcal{P}_{st}} f(p)\big[\gamma_{st}\big(\tau_{st} - t(p)\big) - \alpha^{\text{rep}}(p)\big], \tag{3.11}$$

where $\alpha^{\text{rep}}(p) = \sum_{i\in p} \alpha_i^{\text{rep}}$.

In Equation 3.11, if $\gamma^{st}\big(\tau_{st} - t(p)\big) - \alpha^{\text{rep}}(p)$ is negative, then $f(p)$ is null (objective function decreases otherwise). Hence, from Equation 3.11, we have:

$$t(p) + \frac{\alpha^{\text{rep}}(p)}{\gamma_{st}} \geq \tau_{st} \implies f(p) = 0. \tag{3.12}$$

Since Equation 3.12 is a weight on the logical paths (resulting from the weights of the logical links) that depends only on the offloading overlay, we can narrow the search space of the paths that do not satisfy Equation 3.12. Finally, the formulation can be reduced to the cost-benefit maximization objective subject to the capacity constraint on each logical link $(i,\,j)$:

Maximize $\displaystyle\sum_{s,\,t\in\mathcal{D}} \sum_{p\in\mathcal{P}_{st}} f(p)\psi_{st}^{\text{rep}}(p),$

subject to

$$\sum_{s,\,t\in\mathcal{D}} \sum_{\substack{p\in\mathcal{P}_{st}\\ p\ni(i,\,j)}} \rho_{st}(i,\,j) f(p) \leq c(i,\,j) \qquad \forall (i,\,j) \in L^O,$$

where $\psi_{st}^{\text{rep}}(p) = \gamma_{st}\big(\tau_{st} - t(p)\big) - \alpha^{\text{rep}}(p)$ can be considered as a weight on logical path $p \in \mathcal{P}_{st}$.

3.4.2. The "source replication" model

In this model, data is replicated at the source offloading spot to satisfy the leakage tolerance l_{st} of the offloading demand d_{st}. We assume again that the source can characterize all logical simple paths between the source s and the destination t so it can compute the resulting path leakage $l_p(i,\, j)$ for every logical link $(i,\, j)$ in logical path $p \in \mathcal{P}_{st}$.

We denote by $l_p(i,\, j)$ the multiplied leakage experienced at logical link $(i,\, j) \in L^O$ on logical path p. If $p = (i_0,\, \ldots,\, i_n)$, $i_k \in N^O$ is a logical simple path between i_0 and i_n, then for all $0 \leq k < n$, the leakage $l_p(i_k,\, i_{k+1})$ at logical link $(i_k,\, i_{k+1}) \in L^O$ is expressed as:

$$l_p\,(i_k,\, i_{k+1}) = 1 - \prod_{j=1}^{k} \big(1 - l\,(i_{j-1},\, i_j)\,\big), \tag{3.13}$$

where $0 \leq k < n$. Since the source s knows the attributes of the logical path p, it may deduce the replication factor $\rho_{st}(p)$ introduced in Section 3.3.2. For offloading demand d_{st} allocated on logical path $p = (s,\, i_1,\, \ldots,\, i_{n-1},\, t) \in \mathcal{P}_{st}$, the replication factor $\rho_{st}(p)$ is calculated by the source s as follows:

$$l_p(i_{n-1},\, t)^{\rho_{st}(p)} \leq l_{st} \implies \rho_{st}(p) \geq \frac{\log(l_{st})}{\log(l_p(i_{n-1},\, t))}. \tag{3.14}$$

For each logical link $(i,\, j) \in L^O$, the capacity constraint limits to the logical link capacity $c(i,\, j)$ the amount of assigned flows $f(p)$ for each path p that go through the logical link $(i,\, j)$ with the proportion of leftover flow at the logical link $1 - l_p(i,\, j)$ and replication factor $\rho_{st}(p)$ for offloading demand d_{st}:

$$\sum_{s,\, t \in \mathcal{D}} \sum_{\substack{p \in \mathcal{P}_{st} \\ p \ni (i,\, j)}} \rho_{st}(p)\big(1 - l_p(i,\, j)\big) f(p) \leq c(i,\, j). \tag{3.15}$$

As in the local replication model, the cost-benefit is expressed as the difference of the cost of an Internet-based service (with the cost factor γ_{st}) and the operational cost of the offloading infrastructure $\alpha^{\text{nrep}} \ll \alpha^{\text{rep}}$. In this model, we also aim at maximizing the cost-benefit of offloading data:

$$\sum_{s,\, t \in \mathcal{D}} \left[\beta_{st}\gamma_{st} - \sum_{i \in N^O} \alpha_i^{\text{nrep}} \sum_{j \in N(i)} \sum_{\substack{p \in \mathcal{P}_{st} \\ p \ni (i,\, j)}} \rho_{st}(p)\big(1 - l_p(i,\, j)\big) f(p) \right]. \tag{3.16}$$

If Equation 3.7 is optimal, then it turns into an equality since it is the only equation that constrains β_{st}, given that $\gamma^{st} > 0$. Therefore, we can rewrite Equation 3.16 as follows:

$$\sum_{s,\,t\in\mathcal{D}}\sum_{p\in\mathcal{P}_{st}} f(p)\left[\gamma_{st}\big(\tau_{st}-t(p)\big)-\sum_{\substack{p\in\mathcal{P}_{st}\\ p\ni(i,\,j)}}\alpha_i^{\text{nrep}}\rho_{st}(p)\big(1-l_p(i,\,j)\big)\right]. \tag{3.17}$$

In Equation 3.17, we note that if the factor of $f(p)$ is negative, then, since $f(p) \geq 0$, the objective function decreases, making it non-optimal. Thus, if the factor of $f(p)$ is negative, the flows $f(p)$ must be null. From the former Equation 3.17, we have:

$$t(p) + \frac{\rho_{st}(p)}{\gamma_{st}}\sum_{\substack{p\in\mathcal{P}_{st}\\ p\ni(i,\,j)}}\alpha_i^{\text{nrep}}\big(1-\mathcal{L}_p(i,\,j)\big) \geq \tau_{st} \implies f(p) = 0. \tag{3.18}$$

Similar to Equation 3.12, Equation 3.18 is a weight on the logical paths (although, not resulting from the weights of the logical links) that only depend on the offloading overlay; we can narrow the search space of the paths that do not satisfy Equation 3.18.

Finally, the formulation can be reduced to the total revenue maximization objective subject to the capacity constraint on each logical link $(i,\,j) \in L^O$:

Maximize $\displaystyle\sum_{s,\,t\in\mathcal{D}}\sum_{p\in\mathcal{P}_{st}} f(p)\psi_{st}^{\text{nrep}}(p)$,

subject to

$$\sum_{s,\,t\in\mathcal{D}}\sum_{\substack{p\in\mathcal{P}_{st}\\ p\ni(i,\,j)}}\rho_{st}(p)\big(1-l_p(i,\,j)\big)f(p) \leq c(i,\,j) \qquad \forall(i,\,j)\in L^O,$$

where

$$\psi_{st}^{\text{nrep}}(p) = \gamma_{st}\big(\tau_{st}-t(p)\big) - \sum_{\substack{p\in\mathcal{P}_{st}\\ p\ni(i,\,j)}}\alpha^{\text{rep}}(p)\rho_{st}(p)\big(1-\mathcal{L}_p(i,\,j)\big).$$

The amount of data transferred β_{st} within the delay tolerance τ_{st} imposed by offloading demand d_{st} is deduced from Equation 3.7:

$$\beta_{st} = \sum_{p\in\mathcal{P}_{st}} f(p)\big(\tau_{st}-t(p)\big). \tag{3.19}$$

We can finally obtain from Equation 3.19 the average throughput of offloading demand d_{st} by dividing the amount of data β_{st} by the duration of the transfer τ_{st}.

3.5 | *Performance evaluation on the French road network*

In this section, we evaluate the performance of the revenue maximization problem by comparing the throughput resulting from the cost-benefit maximization under varying parameters, including the Internet-based cost, delay tolerance, leakage tolerance, and logical link leakage.

3.5.1. Dataset of French road traffic

We use a dataset collected in 2011 featuring the Annual Average Daily Traffic (AADT) of the major roads in France covering a combined distance of 20,000 km[4]. The traffic volumes are collected using strategically located automatic traffic recorders. The different thickness and shades of red of the road segments depicted in Figure 3.4 reflect the traffic counts given by the AADTs. The graph consists of 3,310 edges covering over 20,000 km of roads.

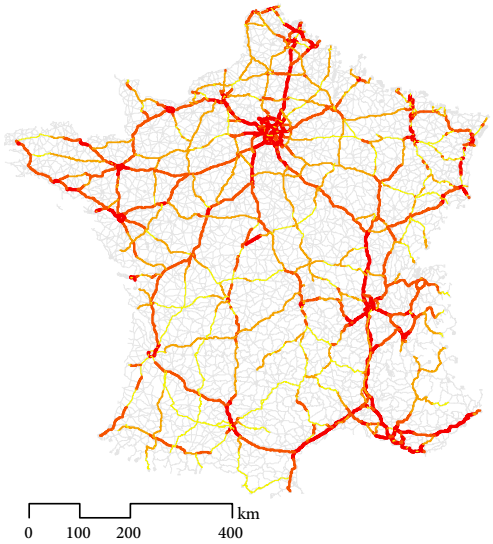


Figure 3.4. Dataset showing the Annual Average Daily Traffic (AADT) of the major roads of France in 2011. The road segments have a darker shade when AADT is higher.

3.5.2. Planning of the network of charging stations

We consider a network of charging stations similar to the one Tesla is currently rolling out in Europe and North America[5]. This network of stations helps electric vehicles face the problem of limited autonomy and achieve long distance travels. Recall that the offloading process will take place in these stations in a transparent manner to the driver. Since there was no such a network in France at the beginning of this thesis, we plan a simple yet realistic network of stations that covers the major roads of France. To plan such a network of stations, we consider a facility-allocation problem that minimizes the number of facilities to allocate, a problem we adapted from the maximal covering location problem [CV74]. The problem takes demand points and candidate locations as inputs:

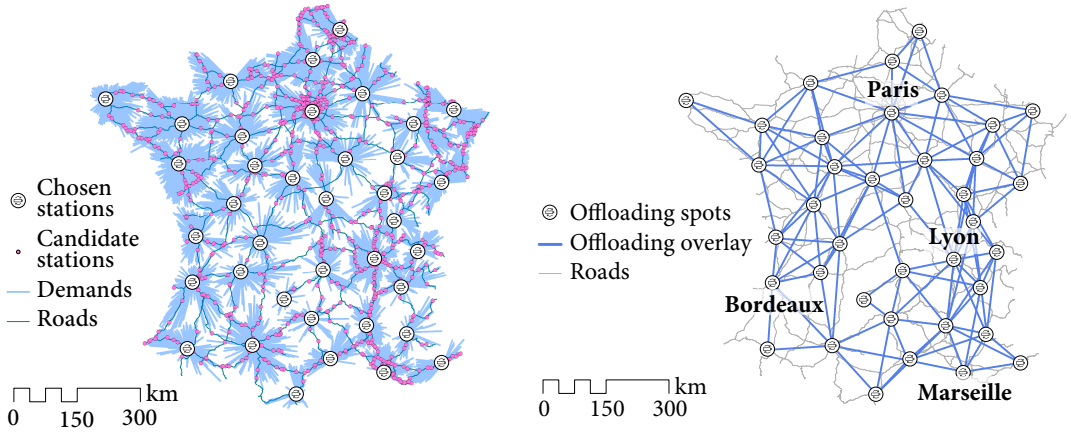
- The demand points are the 9,555 cities of France with a population greater than 1,000.
- The candidate locations are the 1,024 Total gas stations, a French oil company.[6]

The facility-allocation algorithm selects the stations such that a maximum demand points are allocated to the charging stations within a range of 150 km, while minimizing the number of chosen stations. We assume that vehicle ownership is uniform throughout the territory — we can then weight the cities by their population. The chosen stations are allocated at most 150 km away from each other. This is enough for a vehicle with an autonomy of 300 km to reach the next closest station or return to the same station without depleting its battery. Finally, we assume that

[4] ENTD — Census of the road traffic on the French roads in 2011 (in French): `http://tinyurl.com/otfbewv`
[5] `https://www.Teslamotors.com/supercharger`
[6] `http://www.total.fr/mes-deplacements/outils-en-mobilite/export-gps-stations.html` (in French)

the stations have a capacity that suits the demand such it avoids any waiting time at the stations. The waiting time is then restricted to the service time, which corresponds to the duration of the battery charge.

The resulting allocation outputs 38 stations scattered on the French road network, as shown in Figure 3.5a. We note that the stations are mainly allocated near major cities, as the demand from urban areas is higher than from rural areas.



(a) Charging station allocation over the French road network. The big dots are the chosen stations, the small dots are the candidate stations, and the lines represent the allocated demands to the chosen stations.

(b) Resulting offloading overlay built on to of the road network of France. The offloading spots are those represented in Figure 3.5a.

Figure 3.5. Facility-allocation result and offloading overlay. The big dots are the chosen stations.

In the following, we first provide this characterization with conservative assumptions on the road traffic between the offloading spots. Second, we run the revenue maximization problem we introduced in Section 3.3 on the offloading overlay for three large-scale data transfers.

3.5.3. Characterization of the offloading overlay

With the network of stations planned in the previous section, we derive the offloading overlay as defined in Section 3.2.2. To compute the resulting overlay depicted in Figure 3.3, we use the All-or-Nothing traffic assignment strategy that assigns all the road traffic between a source and a destination to the shortest path, here defined as the path with the lowest travel time [dDOW11, She85] (refer to the Appendix A). Note that, this strategy is conservative in terms of road traffic assigned to the road path. The road traffic flow on this path is set to the minimum AADT of the road segments composing the path. Then, we use the results of the most recent French travel household survey made in 2008 (Enquête Nationale Transports et Déplacements (ENTD)) to weight the road traffic on the path by the proportion of trips that travel the length of the path. This survey distinguishes two kinds of travels: local travels are shorter than 80 km from home, while long-distance travels are greater than 80 km from home.

According to the same survey, long-distance travels account for about, in average, 2.8% of the total amount of travels recorded. This kind of travel is of interest when offloading data since

it corresponds to the case where vehicles are likely to stop at more than two offloading spots to charge their depleted batteries while on a trip. We assume the proportion of long-distance travels is uniform on all the road segments. We consider a market penetration ratio $\mathcal{M}$ of 20% and a storage size $\mathcal{S}$ of one Terabyte assumed to be the same for each electric vehicle. Recall that market penetration ratio corresponds to the proportion of vehicles equipped with storage capabilities and able to participate in the data offloading.

Since the AADT is measured in both directions of a road segment, we divide the measured values by two. To compute the throughput of the logical link, we multiply the resulting flow by $\mathcal{S}$. We use the BPR function (Equation 3.1) to calculate the travel time on each road segment with: $\alpha = 0.15$ minutes, $\beta = 4.0$, the practical capacity of each road segment given by the dataset, and the AADT of the road segment for the assigned volume. We use then Equation 3.3 to deduce the travel time of the logical links using the shortest physical road paths.

3.5.4. Benefits of the offloading overlay

For this evaluation, we enumerate all the possible simple paths in both the offloading overlay and the road network. This evaluation shows the benefits of the offloading overlay in mitigating the complexity of the road network.

We plotted the total number of simple paths bounded by a travel time cutoff in both the road network and the offloading overlay. The plot is represented in Figure 3.6 and shows the number of all simple paths as a function of the travel time cutoff of the enumerated paths. In both cases, the number of simple paths grows exponentially with the travel time cutoff. The number of simple paths is much larger on the road network compared to those enumerated in the offloading overlay. Also, the difference in the number of paths grows exponentially. This exponential growth is the main complexity factor to consider when solving the vehicle flow allocation problem.

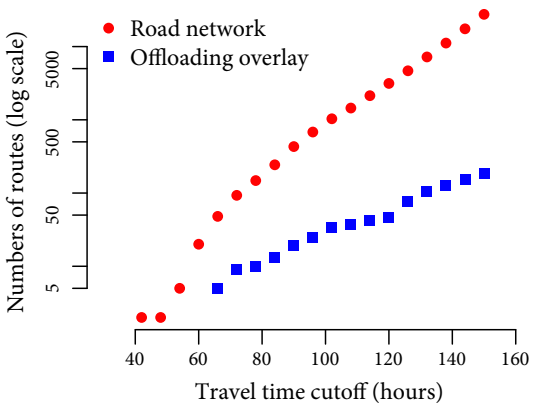


Figure 3.6. Total number of simple paths in the road network and in the offloading overlay as a function of the travel time cutoff (y-axis is in a logarithmic scale).

3.5.5. Revenue maximization of offloading data

We interface the offloading overlay we created with CPLEX using the linear optimization models we presented in Section 3.3. We consider a scenario with the following three different offloading demands (distances are Euclidean) we represent in Figure 3.7:

- Offloading demand d_A: from Paris to Lyon (384 km).
- Offloading demand d_B: from Paris to Bordeaux (492 km).
- Offloading demand d_C: from Paris to Marseille (646 km).

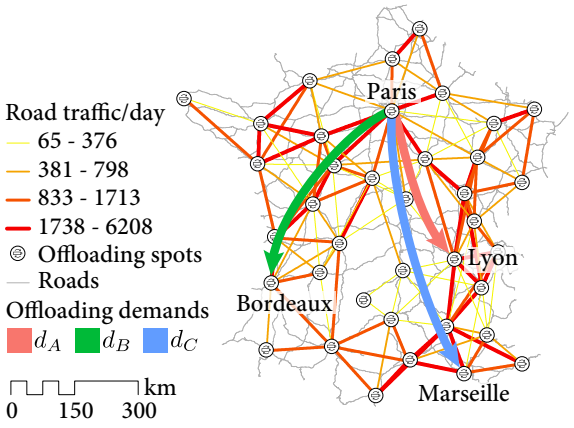


Figure 3.7. Schematic representation of the demands d_A, d_B, and d_C.

It is important to note that demands d_A and d_C will compete for the flows since they share some common subpaths, as we will see in Figure 3.10. This leads to fairness issues, as competing demands will be favored over other demands with the cost-benefit maximization objective, depending on how well they perform. In the following, we discuss how to ensure fairness among the competing flows.

We use a breadth-first search algorithm to generate all the simple logical paths on the offloading overlay for each offloading demand. The cutoffs of the simple paths are given by Equations 3.12 and 3.18 for the local replication *rep* and source replication *nrep* models, respectively. However, the generation of the simple logical paths between a source and a destination is exponential, as seen in Figure 3.6 of the previous chapter. To solve this issue, we reduce our search space by applying a default cutoff of 12 hours on the travel time of the simple logical paths we generate for our experiments. A 12-hour cutoff is sufficient for each generated path to cover all end-to-end trips within France and small enough to avoid unnecessary path computation.

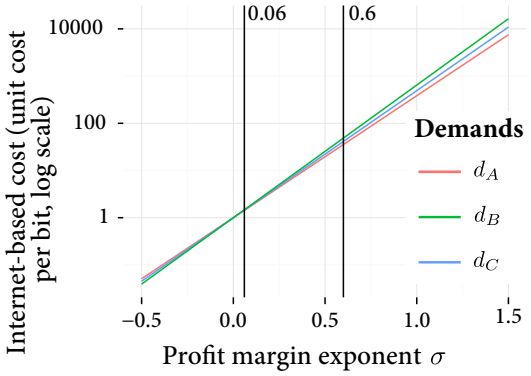


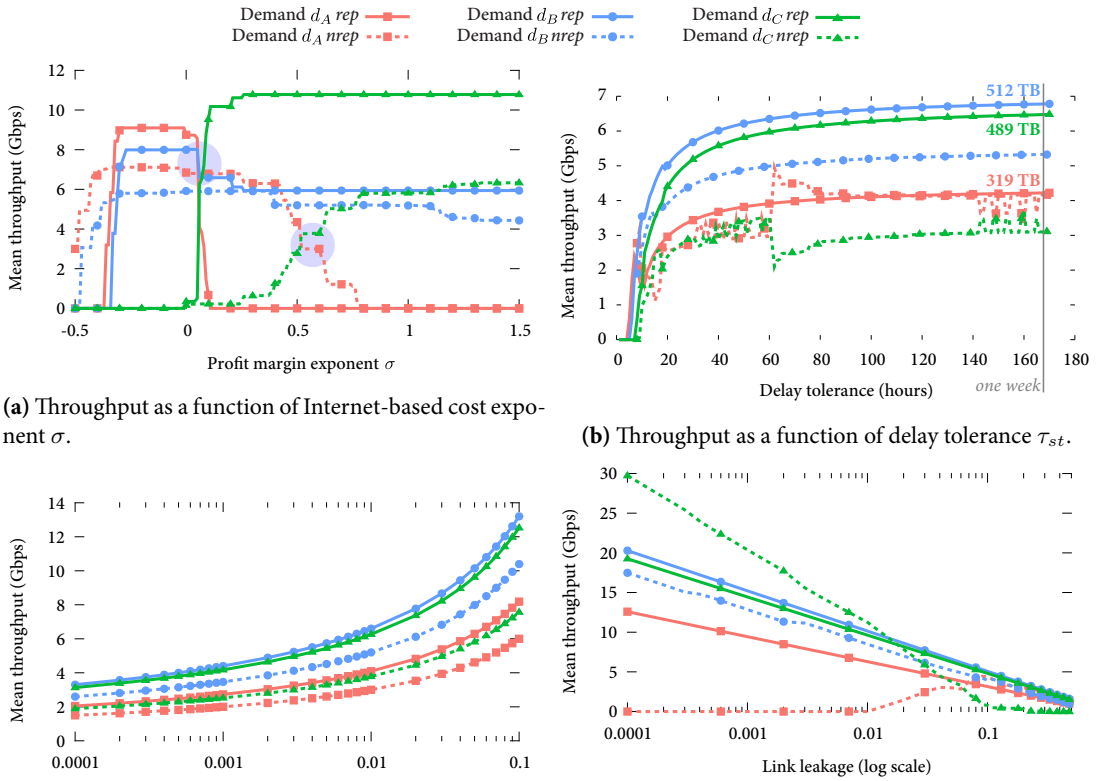
Figure 3.8. Internet-based cost (expressed as the unit cost per bit transferred) as a function of the profit margin exponent for each demand d_A, d_B, and d_C.

We arbitrarily express the Internet-based cost γ_{st} as an exponential function of dist$(s,\ t)$, the distance (in kilometers) between s and t:

$$\gamma_{st} = \left[\text{dist}(s,\ t)\right]^{\sigma},\ \text{where}\ \sigma \in \mathbb{R}. \tag{3.20}$$

We represent the Internet-based cost γ_{st} to transfer a bit of data as a function of the profit margin exponent σ for each demand $d_{st} \in \{d_A,\ d_B,\ d_C\}$ in Figure 3.8. As shown in the figure, the Internet-based cost increases with the distance (*e.g.*, it is greater for the furthest demand d_C) for positive values of the profit margin exponent. Note that the Internet-based cost can be adjusted with a factor to account for more realistic prices.

In our experiments, we investigate the impact of the following parameters: the Internet-based cost exponent σ, the delay tolerance τ_{st}, the leakage tolerance l_{st}, and the logical link leakage $l(i,\ j)$ for logical link $(i,\ j) \in L^O$. By default, for all the demands, we set leakage tolerance to 10^{-2}, logical link leakage to 0.05, and delay tolerance to 96 hours (4 days), such that there is enough time for the allocated paths and flows to stabilize (*i.e.*, in steady state). The operational costs α_i^{rep} and α_i^{nrep} are weighted by the demands allocated to offloading spot i, resulting from the facility-allocation problem. For offloading spot i, we set α_i^{nrep} as the normalized weight of the demands allocated to the offloading spot and $\alpha_i^{\text{rep}} = 1,000 \times \alpha_i^{\text{nrep}}$. Finally, for each of-



(a) Throughput as a function of Internet-based cost exponent σ.

(b) Throughput as a function of delay tolerance τ_{st}.

(c) Throughput as a function of leakage tolerance l_{st}.

(d) Throughput as a function of logical link leakage $l(i, j)$.

Figure 3.9. Evaluation results with, by default, delay tolerance $\tau_{st} = 96$ hours, leakage tolerance $l_{st} = 0.01$, logical link leakage $l(i, j) = 0.05$ for all offloading demands d_{st}. The values of the Internet-based cost σ^{rep} and σ^{nrep} are chosen such that the standard deviation of the throughput of all flows is minimized, to ensure fairness among competing data offloading demands.

floading spot i, we consider a waiting time $\delta_i = 20$ minutes, which corresponds to the duration of a charge that provides a 300 km range to the vehicles (without any queuing time).

As our objective is to obtain fair flow allocation, we tune the Internet-based cost exponent σ accordingly. The arbitrary choice of σ clearly impacts the resulting cost-benefit that can be achieved. That said, different σ values lead to different settings for the parameters we consider (delay tolerance, leakage tolerance, and logical link leakage).

We solve the transfer assignment problem from a macroscopic point of view. In this way, we do not have to consider the scheduling problem and the forwarding decisions made at the offloading spots, which both require an analysis on a per-vehicle basis. The results presented in this section are thus achieved under ideal conditions and show an upper-bound of the performance of the offloading. However, to provide more realistic results, we take account of the scheduling and forwarding errors by means of the logical link leakage.

In Figure 3.9, we first note that the *rep* model (with solid lines) always achieves better throughputs than the *nrep* model (with dashed lines). Indeed, the *nrep* model takes into account all

logical link leakages in the paths it considers and replicates the data accordingly, sending more data than the *rep* model. We also note that our offloading infrastructure can achieve transfers with a throughput above 10 Gbps and aggregated data transfers in the Petabyte range with the *rep* model within a week (*i.e.*, as shown in Figure 3.9b: 512 TB transferred within a week for demand d_B, 489 TB for d_C, and 319 TB for d_A amounts to 1,320 TB transferred within a week).

Figure 3.9a represents the evolution of the throughput of the three demands as a function of σ, given the delay tolerance, the leakage tolerance, and the logical link leakage. We note that the value of σ is decisive concerning the allocation of the flows to favor: either short distance (demand d_A) or long distance (demands d_B and d_C). This behavior happens in both *rep* and *nrep* models, although the breaking points (represented by circles in the plot) are not the same for the two models: the break happens at $\sigma = 0.06$ for the *rep* model, and $\sigma = 0.6$ for the *nrep* model. Beyond these points, long-distance demands are favored compared with short-distance ones. Indeed, γ_{st} is a factor in the maximization objective (defined in Section 3.4) and the higher the γ_{st}, the bigger the total revenue we aim to maximize. As shown in Figure 3.8, since σ is the exponent of the Euclidean distance between s and t, we have:

- If $\sigma < 0$, γ_{st} decreases when the distance increases, favoring short-distance offloading demands.
- If $\sigma = 0$, $\gamma_{st} = 1$, favoring short-distance demands with a lower average travel time between the source and destination.
- If $\sigma > 0$, γ_{st} increases with the distance, favoring long-distance offloading demands.

Although the impact of σ on the allocation of the flows between demands d_A and d_C is strong, the impact is weaker with demand d_B, as this latter shares few subpaths with d_A and d_C, as represented in Figure 3.10. Since we want a fair flow allocation when offloading data, we choose σ^{rep} and σ^{nrep} such that the standard deviation of the throughput of all flows is minimized.

It is important to keep in mind, however, that this allocation has an impact on the total revenue because γ_{st} is the factor that generates the revenue. Since $\sigma^{\text{rep}} < \sigma^{\text{nrep}}$, the *nrep* model has a revenue that is much larger than the *rep* model.

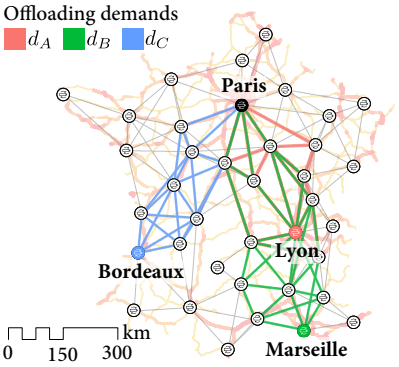


Figure 3.10. Representation of the logical paths allocated to the three demands d_A, d_B, and d_C.

Figure 3.9b shows the throughput as a function of the delay tolerance τ_{st} (*i.e.*, the total duration of the transfer). We notice that the throughput stabilizes as the duration of the transfer increases. For the *rep* model (solid lines), demands with longer distances are favored (demand d_B and d_C) over demands with shorter distances (demand d_A) Indeed, the total revenue is increased when favoring demands with longer distances, as seen on Figure 3.9a. For the *nrep* model (dashed lines), the allocation of demands d_A and d_C oscillates, favoring one or the other. Both demands have flows allocated on shared subpaths, depending on the maximization of the total revenue. The oscillation results from the variation of the value of σ^{nrep}, which depends on the minimization of the standard deviation of the throughput of the allocated flows.

We represent the offloading revenue (in terms of cost unit) for the allocations of demands d_A, d_B, and d_C (respective to Figure 3.9b as a function of the delay tolerance of the offloading demands. Note that the revenue depends on the Internet-based cost and the profit margin exponent σ. Recall that the latter is chosen for each allocation of the demands given a delay tolerance to guarantee fair allocation of the demands, such that it minimizes the standard deviation of the mean throughput resulting from the allocation of the demands. For this reason, the resulting offloading revenue is unstable for small delay tolerances (*i.e.*, < 50 hours). When increasing the delay tolerance of the offloading demands, the profit margin decreases to guarantee the fairness of the allocations of the demands and the revenue increases. Since more data is transferred, the revenue is becoming less sensitive to the transfer distances and σ decreases. Additionally, the revenue resulting from the *rep* model is less than the *nrep* model, as the operational costs when traversing the offloading spots are 1,000 times less for the *nrep* model (since it requires less data to be stored temporarily due to replications).

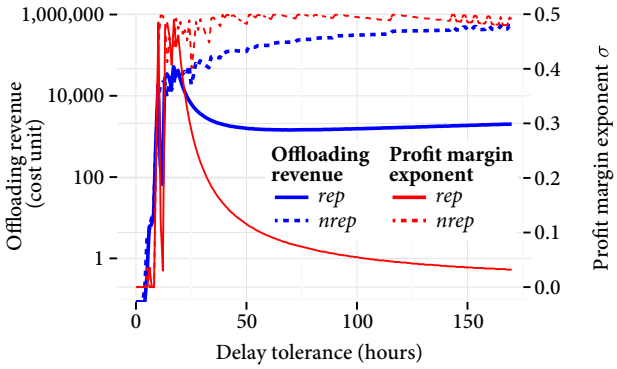


Figure 3.11. Offloading revenue as a function of the delay tolerance resulting from the allocation of demands d_A, d_B, and d_C, respective to Figure 3.9b.

Finally, Figures 3.9c and 3.9d show the throughput as a function of, respectively, the leakage tolerance l_{st} and logical link leakage $l(i,\ j)$. The throughput increases when the leakage tolerance increases or the logical link leakage decreases. The *nrep* model almost achieves the same performance, if not better, than the *rep* model when there is a low logical link leakage ($l(i,\ j) < 0.05$). The logical link leakage also has an effect on the allocation of the flows for the *nrep* model when the flows share subpaths, contrary to the *rep* model: shorter distances (demand d_A) are favored with a high logical link leakage ($\mathcal{L}(i,\ j) > 0.05$) and longer distances (demands d_B and d_C) are favored with a low logical link leakage ($l(i,\ j) < 0.05$). Indeed, the objective function of the *nrep* model depends on the multiplied logical link leakage $l_p(i,\ j)$.

3.6 | *Discussion*

3.6.1. Data security

The owners of the electric vehicles may try to access the data payload they are carrying. Therefore, the data has to be encrypted so that only the content provider is able to read it. For instance, a public-key infrastructure can be considered (*e.g.*, RSA, ECC). The data is encrypted with the public key of the remote entity (the destination of the data) by the originating entity (the source of the data) and will be in turn decrypted by the remote entity using their private key. Also, we can use certificates to authenticate the originating entity of the data by the remote entity.

Also, we consider a multiple path allocation of the data both on the road infrastructure and on the offloading overlay, which offers path diversity. Therefore, the data that is part of the same flow (or even the same data when replicated) will not take the same path (either road path or logical path). This provides robustness for data transfers, as well as increased security since the data belonging to the same flow will be scattered over multiple paths, making it difficult to attack the whole data transfer by preventing single vantage point attacks.

3.6.2. Drayage system

Recall that data is offloaded from a conventional data network to the closest offloading spot using a drayage system. The drayage is equivalent to the first and last mile of the access networks in an Internet-based transfer. The drayage can be carried out by different means:

- *Dedicated lines* (*e.g.*, optical fiber channels) can be set up between the different locations that need drayage. This solution would be adapted to connect locations that require continuous large data exchanges. However, it does not fit temporary data exchanges, as it is a costly solution (*e.g.*, the sources and destinations of a data transfer are only temporary).
- *Dedicated vehicles* can provide data drayage between the different locations. These vehicles would be equipped with storage and communication capabilities, with the data size and rates greater in magnitude than the common vehicles we consider in this thesis.

3.6.3. Predicting the future direction of the stopping vehicles and privacy concerns

For each stopping vehicle, the offloading spot must determine the subsequent offloading spot on the vehicle's route. The offloading service provider stores *anonymously* the previous locations of the vehicles in a historical database to help the offloading spots predict the remaining itinerary of the stopping vehicles. To this end, the service provider can use probabilistic tools, such as Hidden Markov Models [SBZS06], maximum entropy [ZMBD08], or Bayesian networks [LPFK07, KH06]. The partial trajectories of the vehicles can be known through the successive locations recorded by the navigation system of the vehicles. Note that, the current road traffic in the vicinity of the offloading spot can also help predict the most likely routes vehicles will take [XLZL09].

In order to *accurately* predict the future offloading spot the vehicle will visit, the offloading service requires access to the positioning data of the vehicles, which raises some privacy concerns. While it is common for car manufacturers to collect and analyze such data (*e.g.*, Tesla collects the current location of the vehicles for remote vehicle analysis[7]), we are aware of the privacy breach this represents for both the driver and passengers. In Section 3.1.2, we presented a "get paid to drive" program offered by the offloading service provider to the vehicle owners in exchange for transporting a storage device. With this incentive, some drivers may be more willing to share their positioning data with the service. Otherwise, the offloading service provider could use the historical visit of the vehicle at charging stations (operated by the charging station operator) to infer the vehicles' future visits based on probabilistic tools (*e.g.*, Markov chains).

[7] https://www.tesla.com/about/legal

When the drivers do not want to share any data at all with the offloading service, the offloading spots discard the vehicles that do not share their positioning data because there is no means to predict where they will go next. In this case, these vehicles should not be involved in the data offloading.

3.7 | Conclusion

In this chapter, we presented the concept of offloading data traffic onto the road network. This concept exploits the existing mobility of vehicles to extend the capacity of conventional data networks such as the Internet. We assessed this concept by comparing the cost of using the road network to transfer data with the cost of transferring the same amount of data on the Internet. To determine the road path providing similar performance in terms of delay and bandwidth as in the case of the Internet, we proposed an allocation model that maximizes the cost-benefit of offloading traffic on the road network over conventional data transfers. To solve this model in reasonable computational time, we proposed a road map reduction procedure which produces a simplified logical representation of the road network. We evaluated this allocation model on the main roads of France using actual road traffic counts. Our results show that the road network can accommodate simultaneous data transfers with an aggregate capacity in the Petabyte range per week.

4 Centralized control of the offloading infrastructure

In this chapter, we propose an architecture implementing the vehicular data offloading. We leverage the advantages of the logical centralization provided by Software-Defined Networking (SDN) to enable efficient control and management of the offloading infrastructure. SDN provides the necessary logistics including planning, implementing, and controlling for the effective and efficient transportation of data over the road network.

Our SDN architecture consists of a central controller and a collection of offloading spots acting as forwarding engines. Recall that the offloading spots temporarily store data cargo unloaded from the vehicles before loading the cargo to vehicles heading towards the destination of the data. The controller receives demands to offload data transfers onto the road network. The controller is in charge of mapping data transfers onto road network paths and their corresponding flows of vehicles. A path consists of the successive flows of vehicles connecting a sequence of offloading spots. The controller determines the paths by solving the vehicle flow allocation problem we formulate as a max-min fairness model. This formulation maximizes the resulting throughput, while guaranteeing fair allocations of the data transfers to avoid starving data transfers. The output of the vehicle flow allocation problem is then used to configure the sequence of offloading spots involved in the data transfers. Finally, the controller ensures the reliability of data once offloaded by using redundancy and retransmission mechanisms. The controller can thus mitigate the effects resulting from vehicles failing in delivering the data to the next offloading spot.

To solve the vehicle flow allocation problem in a reasonable computational time, the controller leverages the offloading overlay, which provides a logical representation of the offloading infrastructure resulting from a road map reduction procedure. This procedure reduces the high degree of complexity of the road network topology and the large number of vehicular trips.

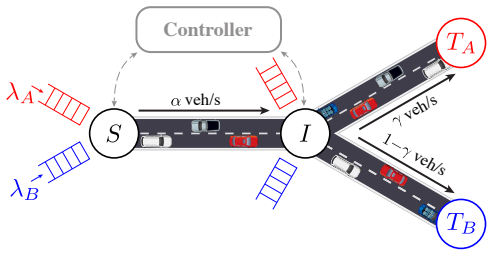
Our main contributions in this chapter are:

- **Centralized architecture** (Section 4.2). We present a centralized architecture that enables scalable and adaptive control of the road network to offload traffic. This architecture also provides the control for efficient and reliable data transfers over the road network.
- **Max-min fair vehicle flow allocation** (Section 4.3). We design an allocation procedure that selects the flows of vehicles matching the performance requirements of an offloading demand while maximizing the throughput of the data carried by those vehicles and guaranteeing fairness among the other demands.
- **Discrete-time evaluation** (Section 4.4). We evaluate our allocation procedure for offloading demands assigned on the French road network using actual road traffic counts.

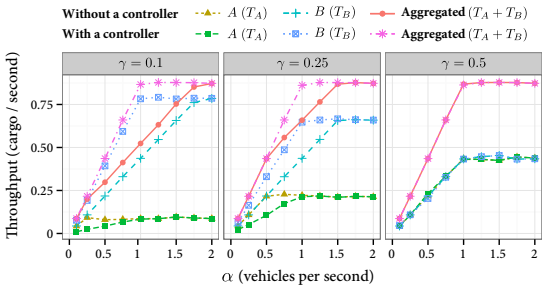
This chapter is structured as follows. As a preamble to this chapter, we give a simple demonstration of the benefits of a centralized control over the offloading infrastructure in Section 4.1. In Section 4.2, we present a centralized architecture that draws on the SDN paradigm to control the offloading infrastructure. In Section 4.3, we present the vehicle flow allocation problem that we formulate using a max-min fairness model. Finally, we evaluate the model using actual road traffic counts for the French road network in Section 4.4, and Section 4.6 draws the conclusions.

4.1 | *Motivating a centralized control*

To assess the benefits of a centralized control of the offloading infrastructure, we use a simple scenario composed of four offloading spots as depicted in Figure 4.1a. We compare different scheduling strategies when concurrent data transfers traverse an offloading spot. Two data transfers A and B originate from offloading spot S and share the road segment connecting offloading spot I. The transfers then follow their route toward their final destinations T_A and T_B, respectively. We only consider the road traffic on the road segments shown in the figure.



(a) Two data transfers A and B originating from offloading spot S share the road connecting offloading spot I. They then follow their own route toward their final destinations T_A and T_B, respectively.



(b) Maximum throughput (in number of data cargo received per second), for the two strategies with and without controller as a function of parameters (α, γ).

Figure 4.1. Road network consisting in four offloading spots to show the benefits of having a controller.

We consider two scheduling strategies, without a controller and with a controller, to select the data transfer and to determine the amount of data to load on the vehicles stopping at each offloading spot. The first scheduling strategy (without a controller) selects the data cargo available at the offloading spots in a round-robin fashion. The data cargo of the transfers is selected in equal proportions and in circular order, without any priority given to the transfers. In the second strategy (with a controller), the decision to load a cargo on a vehicle is based on a predefined configuration of the offloading spots. The controller determines this configuration using the traffic volumes of road network under consideration. In the example of Figure 4.1a, if we assume the road traffic volume $I \rightarrow T_A$ to be three times greater than $I \rightarrow T_B$, the second strategy (with a controller) will load data three times more often onto vehicles heading to T_A.

This is not the case of the strategy without a controller: this strategy will load the same amounts of data on stopping vehicles regardless of whether they head to T_A or T_B.

We use SUMO (Simulation of Urban MObility) to evaluate the three scheduling policies [BBEK11]. We simulate microscopic traffic in volumes of α vehicles per second from S to I, γ from I to T_A, and $1 - \gamma$ from I to T_B. We assume that both transfers A and B have infinite backlog of data at offloading spot S (*i.e.*, $\lambda_A = \lambda_B = \infty$). Figure 4.1b shows the throughput of the system measured in the number of data cargo delivered to the destinations per second. In the scenario, the scheduling is relevant at S, as the road segments connecting the next-hop offloading spot I to T_A and T_B exhibit different road traffic volumes. We consider an infinite backlog traffic generated at S such that all vehicles leaving offloading spot I are loaded with a cargo.

We can see that, for the first strategy without a controller, the transfer headed to T_A needs a longer ramp-up period to reach its nominal throughput. Without any knowledge on the downstream traffic volumes, S cannot load the optimal amounts of data to the next offloading spot I to fully utilize the network resources. As a result, I does not have enough data locally available to feed the higher flow of vehicles traveling towards T_B compared to destination T_A. The resources available on the road segment $(I,\ B)$ remain underused when not enough data belonging to transfer B are transported in an adequate amount to offloading spot I, *e.g.*, for $\alpha < 2$. On the contrary, the second strategy with controller allows S to load data on vehicles traveling to I in adequate amounts for both transfers. In turn, I has enough data to efficiently utilize vehicles traveling on towards T_A as well as T_B. The controller leverages the knowledge of traffic volumes on the downstream road segments to maximize the use of road resources at offloading spot S. This second strategy can achieve a better total throughput performance for all competing transfers at I.

With this simple example, we showed the benefits of a centralized architecture that pre-configures the offloading spots to make the decisions to load data on vehicles. This architecture achieves better throughput compared to a distributed architecture decisions made locally at the offloading spots to load data on vehicles. In the following, we introduce a centralized architecture that controls a large-scale offloading infrastructure.

4.2 | *Centralized controlled offloading architecture*

We exploit the logical centralization provided by Software-Defined Networking (SDN) to enable efficient control and management of the road network resources to offload bulk delay-tolerant traffic from a conventional data network. Following SDN's original design, our architecture consists of two components, as depicted in Figure 4.2: a central controller and the offloading spots acting as forwarding engines. We describe the function of each component in the remainder of this section.

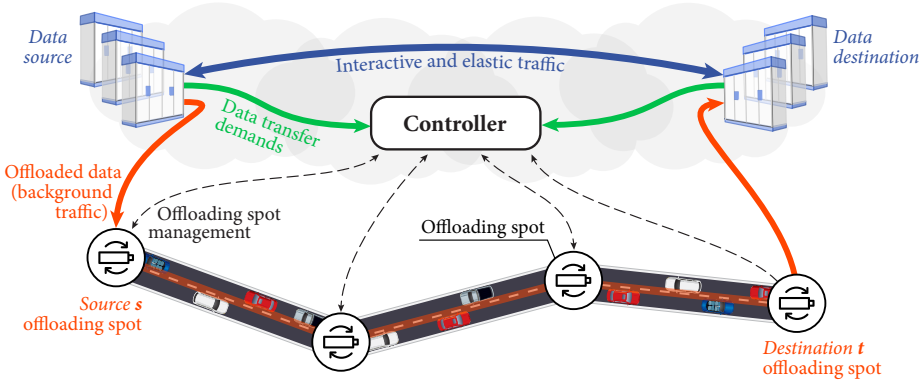


Figure 4.2. Centralized architecture to control the infrastructure in charge of offloading data.

4.2.1. Controller

The controller leverages the offloading overlay we presented in the previous chapter 3.2. As a result, the controller has a holistic view of the road network including its topology and dynamics, such as the traffic volumes for each road segment. Such knowledge may be derived from traffic forecasting services such as Navteq/Here[1], TomTom[2], or Airsage.[3]. The controller keeps track of the status of the offloading spots via a long range control channel (*e.g.*, SigFox[4] or LoRa[5]) [ADTB15]. The information about the offloading spots includes the metadata of the data in transshipment and the statistics about the stopping vehicles, including the historical locations' made available via the navigation system of the vehicles. By collecting the information about the offloading spots, the controller has an up-to-date view of the offloading system. Recall that, in Section 3.6, we presented probabilistic tools to infer the remaining route of a vehicle knowing its navigation history, and the realization of the drayage system.

The controller receives demands to offload data from transfers on the road network. Each demand specifies the delay and bandwidth requirements for the corresponding data transfer, as well as the origin and destination of the transfer. When receiving an offloading demand, the

[1] https://www.here.com/business/traffic
[2] http://automotive.tomtom.com/en/connected-services/tomtom-traffic
[3] http://www.airsage.com/Products/Traffic-Insights/
[4] http://www.sigfox.com/
[5] https://www.lora-alliance.org/

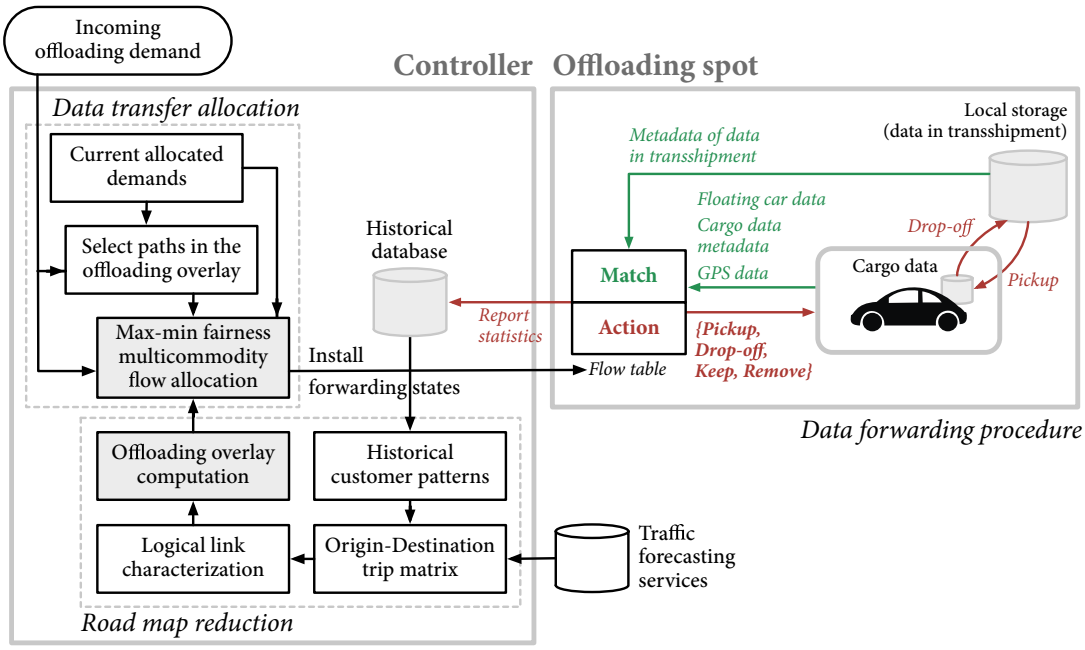


Figure 4.3. Interactions between the functions of the controller and those of the offloading spots.

controller selects the road network paths by solving the vehicle flow allocation problem. A road network path consists of a sequence of offloading spots followed by the data carried by vehicles between the offloading spots. In Section 4.3, we formulate the vehicle flow allocation problem as a Linear Programing (LP) model that maximizes the throughput allocated in a fair manner to the flows of vehicles traveling between the consecutive offloading spots along the road network paths.

Figure 4.3 shows the interactions between the functions of the controller with those of the offloading spots. The allocation procedure involves a reduction of the road network with the offloading overlay we detail in the next section.

4.2.2. Data forwarding at the offloading spots

We represent the interactions between the actions of the controller and the offloading spots in Figure 4.3. The controller installs forwarding states to control the local decisions made at the offloading spots to either drop off or pick up the data cargo. This decision results from the matching of the direction of the stopping vehicles against the destination of the available data. The local decisions account for the allocation of the road network resources shared among concurrent offloaded data transfers.

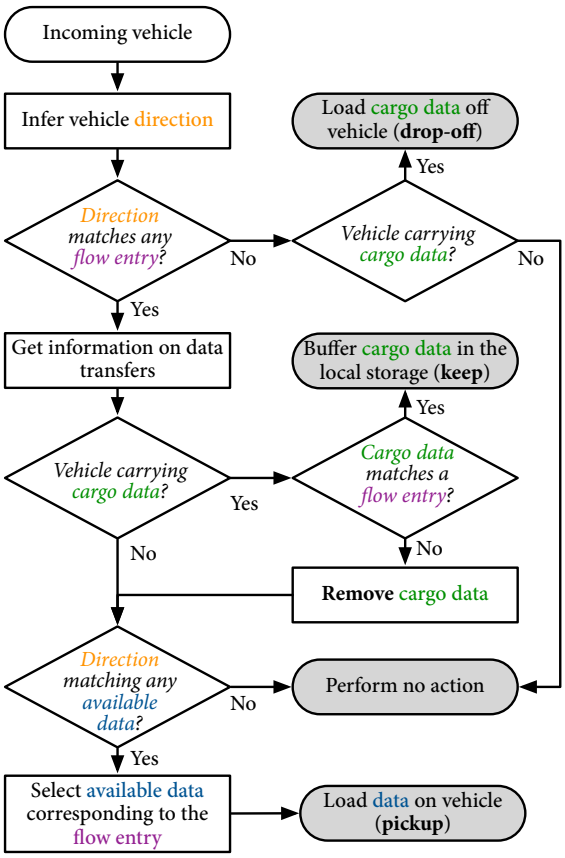


Figure 4.4. Forwarding process at an offloading spot.

Flow tables. The forwarding behavior of an offloading spot is determined by its *flow table* represented in Figure 4.3. It consists of a list of entries, each installed for an individual offloading demands. The controller adds a new entry in the flow table of all the offloading spots on the road network path resulting from the allocation of a data transfer. A flow entry contains the next-hop offloading spot to forward the data cargo along the allocated road path toward its destination. The destination of the stopping vehicles is matched against the next-hop offloading spot of the flow entries. If there is a match, the offloading spot performs an action described by the forwarding process.

Forwarding process. The forwarding process is represented by the flowchart depicted in Figure 4.4. Upon the arrival of a vehicle, an offloading spot checks if the direction of the vehicle matches one entry of its flow table. If none of the entries match, the vehicle unloads, if any, its data cargo onto the offloading spot storage for future pick-ups and continues its route without performing any further actions. If multiple entries match the direction of a vehicle, the offloading spot selects one entry based on the scheduling strategies presented in Section 4.3.3. If the vehicle already carries data, the offloading spot checks if this data belongs to the data transfer represented by the matching entry. If this is the case, the vehicle keeps its cargo and continues its journey. Otherwise, the vehicle unloads its cargo at the offloading spots and becomes empty. For an empty vehicle, the oldest cargo associated with the selected entry is transferred to the vehicle.

4.3 | *Max-min fair vehicle flow allocation model*

In this section, we introduce the max-min fair allocation model solved by the controller in response to offloading demands. This model maximizes the throughput of the data flows allocated in a fair manner to the flows of vehicles traveling between the offloading spots. This model also integrates redundancy and retransmission mechanisms to completely recover from the effects of the logical link leakage. We then present the scheduling policies used by an offloading spot to serve concurrent data transfers.

We consider the same assumptions regarding the offloading demands as Section 3.3.1 of the previous chapter. The demand is characterized by the amount of data β_{st} and the deadline τ_{st} before which the transfer should be completed. However, here, we do not characterize the

offloading demands with the leakage tolerance, since we target reliable transfers that do not tolerate any data losses. For simplicity, we model the rate of the demands at s by a Poisson distribution λ_{st} and its mean value is the average throughput β_{st}/τ_{st}.

To improve the readability, we list the notations we use in the rest of this chapter in Table 4.1.

Table 4.1: Table of notations for the max-min fair allocation model.

Variable	Meaning
d_{st}	Offloading demand between source s and destination t
τ_{st}	Deadline to transfer the data of offloading demand d_{st}
β_{st}	Amount of data to transfer for offloading demand d_{st}
λ_{st}	Poisson arrival rate at the source for offloading demand d_{st}
$\mathcal{P}_{st}$	Set of simple paths between s and t on the offloading overlay
$\mathcal{S}$	Storage capacity of the vehicles (cargo size)
$\mathcal{M}$	Market penetration ratio
$c\,(i,\,j)$	Capacity of logical link $(i,\,j)$
$t\,(i,\,j)$	Travel time on logical link $(i,\,j)$
$l\,(i,\,j)$	Leakage of logical link $(i,\,j)$
$l_{st}^{\textbf{red}}\,(i,\,j)$	Leakage of logical link $(i,\,j)$ with redundancy for offloading demand d_{st}
$o_{st}^{\textbf{red}}$	Weight of the **red**undancy mechanism on the data flow for offloading demand d_{st}
$R_{st}^{\{\text{hbh,e2e}\}}(i,\,j)$	Average number of transmissions (either end-to-end (**e2e**) or hop-by-hop (**hbh**) of a data cargo on logical link $(i,\,j)$ for offloading demand d_{st}
δ_i	Waiting time at offloading spot i (to charge the EV's battery and load data)
$f\,(p)$	Data flow on logical path p
$t\,(p)$	Travel time of logical path p
$O_{st}(i,\,j)$	Overhead of the offloading demand d_{st} on the flow at logical link $(i,\,j)$
$R_{st}(i,\,j)$	Number of transissions of a cargo belonging to demand d_{st} on logical link $(i,\,j)$, regardless of the retransmission mechanism

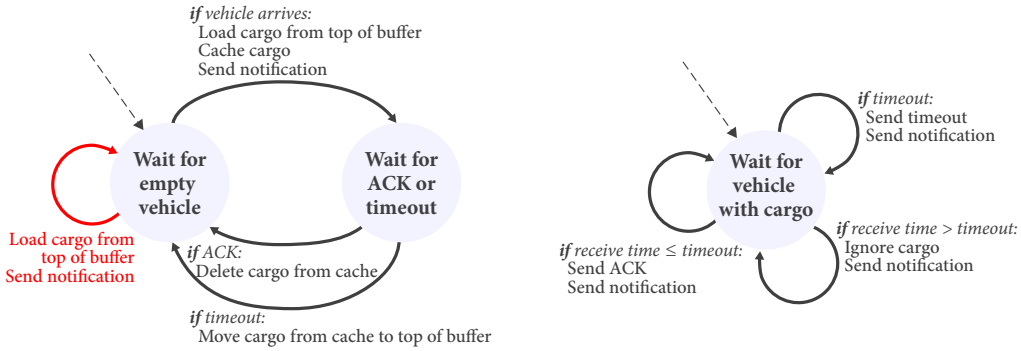
4.3.1. Reliability overhead

In this section, we express the overhead in the throughput resulting from the two reliability mechanisms we consider. Firstly, we express the overhead for the redundancy mechanism in the case of RAID 6. Secondly, we express the overhead for the retransmission mechanism in the case of Selective Repeat Automatic Repeat reQuest (SR-ARQ).

4.3.1.1 RAID redundancy

Without loss of generality, we use RAID 6 to partially recover from losses resulting from a vehicle failing to deliver its data cargo to the next offloading spot. RAID 6 divides the β^{st} data of an offloading demand d_{st} into N arrays of $n \geq 4$ cargo. An array consists of two redundant cargo for $n-2$ cargo payloads. Therefore, a data transfer of N RAID 6 arrays requires nN vehicles, including $2N$ vehicles carrying redundant cargo to recover the losses in the $N(n-2)$ other vehicles.

RAID redundancy increases the amount of data sent by a factor o_{st}^{red}. For a data transfer involving exactly n data cargo arranged in N arrays, we express o_{st}^{red} as follows:



(a) Sender-side (offloading spot i).

(b) Receiver-side (offloading spot j).

Figure 4.5. SR-ARQ retransmissions. The red transitions is exclusively for the intermediate offloading spots in case of end-to-end retransmissions. The other transitions is for the source offloading spot in case of end-to-end retransmission and the source and the intermediate offloading spots in case of hop-by-hop retransmission.

$$o_{st}^{\text{red}} = \frac{n}{n-2}. \tag{4.1}$$

The data carried by n vehicles whose storage devices are arranged in RAID 6 and travelling the logical link $(i,\ j)$ experiences a reduced data leakage denoted $l_{st}^{\text{red}}(i,\ j)$. We express $l_{st}^{\text{red}}(i,\ j)$ in terms of $l(i,\ j)$ (*i.e.*, the data leakage $(i,\ j)$ without data redundancy protection) as follows:

$$l_{st}^{\text{red}}(i,\ j) = 1 - \sum_{k=0}^{2} \binom{n}{k} l(i,\ j)^k \big(1 - l(i,\ j)\big)^{n-k}. \tag{4.2}$$

Note that this expression assumes a data linkage equivalent to the failure likelihood of a storage device, which is consistent as both are independent and identically distributed.

We explain how we determine n (the number of storage devices per array) at the end of this section, as it depends on the total amount of transmitted data (including redundant data and the additional copies introduced by the retransmissions for recovering losses that RAID cannot repair).

4.3.1.2 SR-ARQ retransmissions

In addition to data redundancy, we use Selective Repeat Automatic Repeat reQuest (SR-ARQ) to fully recover the losses that RAID cannot recover [LC04]. We propose two retransmission strategies: hop-by-hop (hbh) retransmissions and end-to-end (e2e) retransmissions. In Figure 4.5, we represent the sender-side in Figure 4.5a and the receiver-side in Figure 4.5b. The red transition in the sender-side refers to the behavior of the intermediate offloading spots in case of end-to-end retransmissions, while the other transitions refer to the behavior of the

source offloading spot in case of end-to-end retransmission and all the offloading spots in case of hop-by-hop retransmissions. The controller enforces the retransmissions at the offloading spots using a low-bandwidth control channel. Both retransmissions strategies rely on timeout expiration at the receiver-side to trigger a retransmission. The timeout corresponds to the travel time $t(i, j) + \varepsilon$ (ε constant) between two offloading spots i and j and depends on the current road traffic. Note that, the controller can also leverage traffic forecasting services to re-evaluate the travel time using exponential weighted moving average techniques.

With the hop-by-hop strategy, the controller is informed of loaded vehicles leaving offloading spot i. A copy of the cargo is cached by i until successfully transmitted to j, the next-hop offloading spot. If no acknowledgment is received from j before the timeout expires, the controller notifies i to retransmit the missing cargo. The hop-by-hop strategy introduces an overhead corresponding to the average number of transmissions $R_{st}^{\text{hbh}}(i,\, j)$ needed to successfully deliver a cargo on logical link $(i,\, j)$. We express $R_{st}^{\text{hbh}}(i,\, j)$ as follows:

$$R_{st}^{\text{hbh}}(i,\, j) = \frac{1}{1 - l_{st}^{\text{red}}(i,\, j)}. \tag{4.3}$$

With the end-to-end strategy, the only offloading spot in charge of recovering a data loss is the first one located at the source-side of the data transfer. Once the last offloading spot on the destination-side receives the data, it notifies the controller of the successful delivery. In turn, the controller notifies the first offloading spot on the source-side to delete the cached copy of the cargo. If the intermediate offloading spots do not receive the expected cargo before the timeout expires, they notify the controller, which in turn asks the source to send another copy of the cargo. The overhead of the end-to-end strategy corresponds to $R_{st}^{\text{e2e}}(i,\, j)$, the average number of transmissions of a cargo on logical link $(i,\, j)$ before successfully delivering it from s to t. We express $R_{st}^{\text{e2e}}(i,\, j)$ as follows:

$$R_{st}^{\text{e2e}}(i,\, j) = \frac{(-1)^{n-(i-1)}}{\prod_{j=i}^{n-1} \left(l_{st}^{\text{red}}(j_i,\, j_{i+1}) - 1\right)}. \tag{4.4}$$

We denote $O_{st}(i,\, j)$ as the total overhead introduced by RAID 6 and SR-ARQ on logical link $(i,\, j)$ to recover from vehicles failing to deliver the data sent with offloading demand d_{st}. We express $O_{st}(i,\, j)$ in terms of the overheads introduced by both the redundancy and retransmission mechanisms as follows:

$$O_{st}(i,\, j) = \begin{cases} o_{st}^{\text{red}} \times R_{st}^{\text{hbh}}(i, j) & \text{(hop-by-hop)} \\ o_{st}^{\text{red}} \times R_{st}^{\text{e2e}}(i, j) & \text{(end-to-end)} \end{cases} \tag{4.5}$$

where o_{st}^{red} is given by Equation 4.1.

In the following, we will use $o_{st}^{\text{ret}}(i,\, j)$ to represent the overhead of the retransmission mechanism regardless of whether the mechanism is hop-by-hop or end-to-end. Similarly, we use $R_{st}(i, j)$ to represent the number of transmissions on logical link (i, j) for cargo belonging to demand d_{st}, regardless of the retransmission mechanism.

4.3.1.3 Determination of the array size n

RAID 6 redundancy distributes the data across arrays of n data cargo. The total number of data cargo n is defined so that to minimize the data overhead O_{st} needed to ensure reliable transfer in response to offloading demand d_{st}:

$$\min O_{st} = \begin{cases} \min\limits_{n} \left\{ o_{st}^{\text{red}} \times \max\limits_{(i,j)\in p} \{R_{st}^{\text{hbh}}(i,j)\} \right\} & \text{(hop-by-hop)} \\ \min\limits_{n} \left\{ o_{st}^{\text{red}} \times \max\limits_{(i,j)\in p} \{R_{st}^{\text{e2e}}(i,j)\} \right\} & \text{(end-to-end)} \end{cases}$$

4.3.2. Max-min vehicle flow allocation model

The controller receives the demands to offload data transfers characterized by their performance requirements. The task of the controller consists in selecting the flows of vehicles according to the following constraints: (*i*) the number of vehicles is sufficient to meet the transfer requirements and (*ii*) the allocation of the combined storage of the vehicles is efficient and fair among the competing transfers. To this end, the controller starts by computing the logical paths in the offloading overlay for each of the transfer to offload. The controller then allocates flows of data on these paths while satisfying both constraints and finally configures then the offloading spots along the selected logical paths with a non-zero allocation.

In the following, $\mathcal{P}_{st}$ denotes the set of candidate logical paths between s and t. Each logical path $p \in \mathcal{P}_{st}$ consists of a sequence of logical links connecting pairs of offloading spots in the offloading overlay. We express the path travel time $t(p)$ experienced by a data transfer allocated to flows of vehicles traveling on logical path p belonging to the set of logical paths $\mathcal{P}_{st}$ for offloading demand d_{st}. The path travel time is the sum of two components: travel time component and waiting component. The travel time component is the sum of the travel time of the logical link of p. The waiting component is the sum of the waiting times at each offloading spot connecting these logical links. We express $t(p)$ as a function of $R_{st}(i,j)$, the average number of transmissions of a cargo to successfully deliver it from i to j for demand d_{st}:

$$t\,(p) = \sum_{\substack{p\in\mathcal{P}_{st} \\ (i,\,j)\in p}} R_{st}(i,\,j)\,[\delta_i + t(i,\,j)]\,, \tag{4.6}$$

where δ_i is the waiting time at offloading spot i.

4.3.2.1 Linear programming formulation

We formulate the vehicle flow allocation procedure as a Linear Programing (LP) model that determines the logical paths matching the performance requirements of the offloading demands. The LP shown in Algorithm 4.1 consists in allocating $f(p)$ data to the flows of vehicles traveling the logical paths of $\mathcal{P}_{st}$. We first present the inputs and then the allocation strategy for the

offloading demands. The strategy relies on the multi-commodity flow allocation problem we formulate as a linear programming model.

Input: Offloading demands $\mathcal{D} = \{d_{st}\}$, characterized by β_{st} and τ_{st}
Set $\{\mathcal{P}_{st}\}_{st}$ of the logical paths between s and t for each demand in $\mathcal{D}$
Average travel time $t(p)$ on logical path p
Capacity $c(i,\, j)$ of logical link $(i,\, j)$
Output: Resulting allocation $A = \{f(p)\}$ of flows to logical paths p

Procedure Allocation **:**
$L \leftarrow \{\text{Logical links } (i,\, j)\}$
$\{f(p)\} \leftarrow$ Max-Min Fairness($\mathcal{D}$, L)
return $\{f(p) : p \in \mathcal{P}_{st},\, (s,\, t) \in \mathcal{D}\}$

Function Max-Min Fairness($\mathcal{D}$, L) **:**
Initialization: $U \leftarrow \mathcal{D};\ i \leftarrow 0;\ A \leftarrow \emptyset$
while $U \neq \emptyset$ **do**
Maximize the i-th smallest allocation:
$\phi_i \leftarrow$ MCF($\mathcal{D}$, C, U, i, A)
Perform non-blocking test:
$A_i \leftarrow$ Non-Blocking Test(U, A, ϕ_i)
Fix the allocation of each demand in A_i to ϕ_i
$U \leftarrow U \backslash A_i$
$i \leftarrow i + 1$
return $\{f(p) : p \in A\}$

Function MCF(D, C, U, i, A) **:**
Maximize $\quad \phi_i$
Subject to:

$$\sum_p f(p)\big(\tau_{st} - t(p)\big) \geq \beta_{st} \qquad \forall (s,\, t) \in \mathcal{D}$$
$$\phi_i - \sum_p f(p)\big(\tau_{st} - t(p)\big) \leq 0 \qquad \forall (s,\, t) \in U$$
$$\phi_k - \sum_p f(p)\big(\tau_{st} - t(p)\big) = 0 \qquad \forall (s,\, t) \in A_k, \phi_k \text{ constant}, \; k = 0,\, \ldots,\, i-1$$
$$\sum_{s,\,t} o_{st}^{\text{red}} \sum_p \left[o_{st}^{\text{ret}}(i,\, j) f(p)\right] \leq c(i,\, j) \qquad \forall (i,\, j) \in L, \; p \text{ s.t. } (i,\, j) \in p$$

return $\{\phi_i\} \cup \{f(p) : p \in \mathcal{P}_{st},\, (s,\, t) \in \mathcal{D}\}$

Algorithm 4.1: Computing the allocations with the max-min fair allocation model.

Inputs. The model takes as input the set $\mathcal{D}$ of all demands to offload a data transfer on the road network. This set includes the previous demands already allocated in addition to the new demands.[6] The allocation procedure also takes as input $\mathcal{P}_{st}$, the set of candidate logical paths between each pair s and t for all demands in $\mathcal{D}$. To enumerate the logical paths in $\mathcal{P}_{st}$, we propose

[6] We have to re-allocate the previous demands to guarantee fair allocation of these demands and the new ones.

to use Yen's k-shortest paths algorithm [Yen71] or a breadth-first search algorithm [Lee61]. In our simulations, we reduce the search space by considering the logical paths sorted according to the travel time of a single cargo. The offloading overlay and the properties of each logical link (*i.e.*, capacity, travel time, and data leakage) are also inputs of the allocation procedure.

Model. The controller allocates $f(p)$ flows to the logical paths listed in $\mathcal{P}_{st}$ according to the Max-Min fairness strategy. This strategy proceeds by successive iterations. The first iteration allocates the minimum flows to satisfy the requirements of the demands (given by the first constraint in the MCF function). The next iterations successively allocate the remaining capacity of the network to the demands that can receive more flow. More specifically, iteration i maximizes the minimal flow allocation noted ϕ_i and fixes the allocation for the demands that cannot be better served, *i.e.*, because of the capacity constraints of the paths or if the demand requirements are already satisfied. The next iterations process the remaining demands. To determine for which transfers the current allocation can be further increased in the following iterations, we use the non-blocking test algorithm presented shown in Algorithm 4.2.

Input: Set U of demands allocated at step i
Set A of demands allocated at steps $k < i$
Allocation ϕ_i of step i
Output: Set A_i of flows in U that cannot be allocated more than ϕ_i in any solution

Function Non-Blocking Test(U, A, ϕ_i) **:**

Demands that have been satisfied are allocated:
$D_i = \{(s,\, t) \text{ s.t. } \sum_p f(p)\big(\tau_{st} - t(p)\big) = \beta_{st}\}$
$U \leftarrow U \backslash D_i;\ A_i = D_i$
Demands allocated more than ϕ_i can be increased:
$U_i \leftarrow \{(s,\, t) \text{ s.t. } \sum_p f(p)\big(\tau_{st} - t(p)\big) > \phi_i\}$
Test the allocation of the remaining demands:
foreach demand $(s,\, t) \in U \backslash (A_i \cup U_i)$ **do**
Solve the following linear program:

Maximize	$\sum_{p \in \mathcal{P}_{st}} f(p)$
Subject to:	
$\sum_p f(p)\big(\tau_{st} - t(p)\big) \geq \beta_{st}$	$\forall (s,\, t) \in D$
$\phi_k - \sum_p f(p)\big(\tau_{st} - t(p)\big) = 0$	$\forall (s,\, t) \in A_k, \phi_k$ constant, $k = 0, \ldots, i-1$
$\sum_{s,\,t} o_{st}^{\text{red}} \sum_p \left[o_{st}^{\text{ret}}(i,\, j) f(p)\right] \leq c(i,\, j)$	$\forall (i,\, j) \in C$, p s.t. $(i,\, j) \in p$

if $\sum_{p \in \mathcal{P}_{st}} f(p) \leq \phi_i$ **then** $A_i \leftarrow A_i \cup \{(s,\, t)\}$
else $U_i \leftarrow U_i \cup \{(s,\, t)\}$
return A_i

Algorithm 4.2: Non-blocking test for the max-min fairness allocation model.

The core of the max-min fairness algorithm is the MCF (MCF) function shown in Figure 4.1. The MCF function computes the multi-commodity flow allocation for the remaining demands. The first constraint matches the amount of data that can be offloaded within the deadline τ_{st} to the amount of data to transfer β_{st}. The following constraint ensures that the demands belonging to the sets A_k, $k \in [0,\, i-1]$ are fixed to the allocation they received at previous step k. The objective of the MCF function is to maximize the minimum allocation ϕ_i such that all demands are satisfied. This objective is further guaranteed by the third constraint of the linear problem. Finally, the last constraint limits the total allocation of the paths crossing the logical links to the logical link capacity. Note that this constraint takes the overhead of the retransmission and the redundancy mechanisms into account.

Once the allocation ϕ_i have been fixed by the MCF function for iteration i, the max-min fairness algorithm determines which demands can be further increased in their current allocation using the non-blocking test algorithm shown in Figure 4.2. The non-blocking test is derived from the algorithm proposed by Pióro *et al.* [PNKF03]. This test compares the maximal throughput of the flows allocated by a multi-commodity flow allocation to the one resulting from the minimal flow allocation ϕ_i. If the multi-commodity flow allocation improves the maximal amount of data allocated to a demand, the demand will be fixed in the next iterations of the max-min fairness algorithm. Otherwise, the demand cannot be better increased, and it is fixed to ϕ_i.

Note that the formulation of the multi-commodity flow remains valid for continuous flows (infinite backlog data traffic). In this case, we ignore the first constraint in the MCF function that satisfies the demand requirements.

The max-min fair allocation model presented in this section has a polynomial time-complexity because fractional flows are allowed in the linear programming models formulated in function MCF and the non-blocking algorithm [Kar08]. The time-complexity of both linear programming models grows in linear time with the number of demands and the number of paths in $\mathcal{P}_{st}$ for each demand d_{st} of the set of demands to allocate.

4.3.3. Data scheduling

Recall from Section 4.2.2 that offloading spots use flow tables to match the data locally available with the stopping vehicles and decide an action to take. Multiple entries in the flow table of an offloading spot i may match the direction of a vehicle. Each entry corresponds to different data transfers or the same transfer spanning many paths in the road network. The offloading spot selects in which order the flow of vehicles should be allocated to the transfers according to the weighted fair queuing scheduling policy whose weights are configured by the controller, resulting from the allocation procedure output. We denote by $f(d_k p_l)$ the data flow on logical path p_l allocated to the transfer resulting from offloading demand d_k. The controller assigns a weight $w(d_k p_l)$ to the transfers allocated to the logical paths in $P = \{d_k p_l | \forall d_k \in D,\, p_l \in \mathcal{P}^{d_k},\, (i,\, j) \in p_l\}$. P is the set of logical paths passing by offloading spot i and sharing the same next-hop offloading spot j. The controller computes $w(d_k p_l)$ by normalizing the rate of

flow $f(d_k p_l)$ with the total rate of the flows traveling all paths in P:

$$w(d_k p_l) = \frac{f(d_k p_l)}{\sum_{p \in P} f(p)}. \tag{4.7}$$

We use the weights in the scheduling algorithm to determine the priority of data transfers to assign the corresponding data cargo to a passing vehicle if multiple transfers traverse the same offloading spot. In our simulations, we considered a probabilistic weighted fair queuing scheduler, which uses the weights to pick a data transfer at random and assign the corresponding cargo to the stopping vehicle [SV96].

4.4 | *Performance evaluation on the French road network*

In our evaluation, we use the same dataset as the one we used in the previous chapter to conduct the evaluation of the throughput resulting from the cost-benefit maximization of offloading data (refer to Section 3.5.5).

We provide a characterization of the offloading overlay introduced in Section 4.4.1 using more comprehensive road traffic modelling techniques compared to Section 3.5.3 to estimate the road traffic volumes between the offloading spots. Secondly, we evaluate the max-min fair allocation model we introduced in Section 4.3.2 using the dataset of the French roads.

4.4.1. Road map reduction

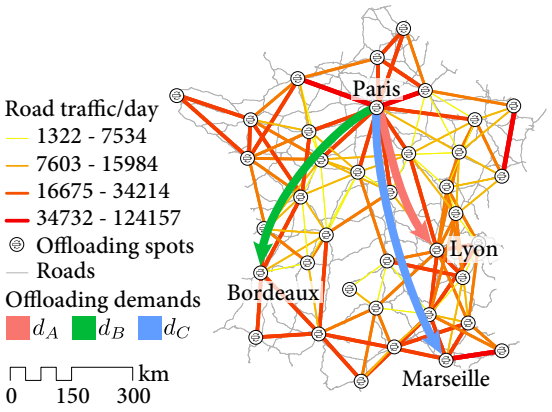


Figure 4.6. Schematic representation of the demands d_A, d_B, and d_C.

In the following, we characterize the logical links of the offloading overlay by expressing the road traffic volumes between adjacent offloading spots in terms of network quantities. These volumes are given by origin-destination matrices between the offloading spots. However, the datasets we use in our simulations only give the traffic counts in terms of AADT (*i.e.*, the number of vehicles per unit of time for each road segment). We discuss the reasons why we use traffic counts in Section 4.5.

We use well-known traffic modelling techniques from transportation research to estimate the origin-destination matrix of the offloading spots from the traffic counts of the dataset. We detail the traffic modelling techniques in the Appendix A. We proceed with the following three steps.

1. *Route determination.* The first step consists in selecting a subset of the alternative routes (to the shortest one) connecting each pair of adjacent offloading spots in the road network. The selection consists in choosing the k-shortest routes in terms of travel time.

The routes are also selected such that they share a low degree of similarity in terms of stretches of road in common. We implement this selection process by using the algorithms proposed by Abraham *et al.* [ADGW13].

2. *Route assignment.* The second step consists in assigning weights to the selected routes using the C-logit route assignment model [CN96]. The value of a weight is determined according to properties such as the travel time and the distance of the route. Those weights reflect the capacity of a route in attracting traffic, the higher the weight of a route the more traffic it will receive. The weights are then used in combination with the traffic counts to estimate the traffic volume of the routes selected in the first step between each pair of adjacent offloading spots.
3. *Trip matrix estimation.* In the third step, we use the entropy maximization model proposed by Zuylen and Willumsen to compute the origin-destination trip matrix consisting in all pairs of offloading spots in the offloading overlay [VZW80]. This model determines the most likely distribution of the traffic across all the routes selected in Step 1 subjected to two constraints, namely the traffic counts of the routes' stretches of road and the C-logit weights. We feed this model with a dataset of the French roads we consider to infer the O-D trip matrix between the adjacent offloading spots.

Finally, we determine the attributes of each logical link $(i,\, j)$ in the offloading overlay. These attributes are relevant to the allocation of the data transfers. The attributes are as follows:

- *Capacity* $c(i,\, j)$. The capacity of $(i,\, j)$ represents the combined storage of all vehicles travelling between i and j. The capacity also reflects the market penetration ratio $\mathcal{M}$, *i.e.*, the ratio of vehicles equipped with data storage devices.
- *Travel time* $t(i,\, j)$. The travel time is computed as the travel time average for each route selected in the first step between i and j and weighted by the route weights computed in the second step.
- *Leakage* $l(i,\, j)$. The leakage represents the ratio of vehicles that fail to deliver the data they transport to the next offloading spot.

4.4.2. Max-min fair allocation model on the roads of France

The objective of the simulation is to evaluate three metrics: (*i*) maximum throughput of the resulting allocation to evaluate the capacity of the offloading infrastructure, (*ii*) delay to transfer pre-defined amounts of data depending on the number of offloading spots involved in the data transfers and (*iii*) fairness of the allocation of concurrent transfers when using logical paths with similar lengths.

We evaluate the performance of the transfers resulting from the allocation of three offloading demands using the max-min fair allocation model. The three demands are shown in Figure 4.6: (*i*) d_A from Paris to Lyon with arrival rate λ_A, (*ii*) d_B from Paris to Bordeaux with arrival rate λ_B, and (*iii*) d_C from Paris to Marseille with arrival rate λ_C. Note that the road paths followed by the transfers resulting from demands d_A and d_C share the same logical links in the offloading network; as so d_A and d_C are competing over those links. We use SUMO to run the simulations,

which each lasts 300,000 seconds (3.5 days), including 43,200 seconds (12 hours) of *warmup*, to give time for the first data cargo to reach their respective destination. In the rest of this section, we consider a conservative market penetration ratio $\mathcal{M}$ of 10%. Data leakage is assumed to be the same for all logical links in the offloading overlay.

4.4.2.1 Maximum throughput

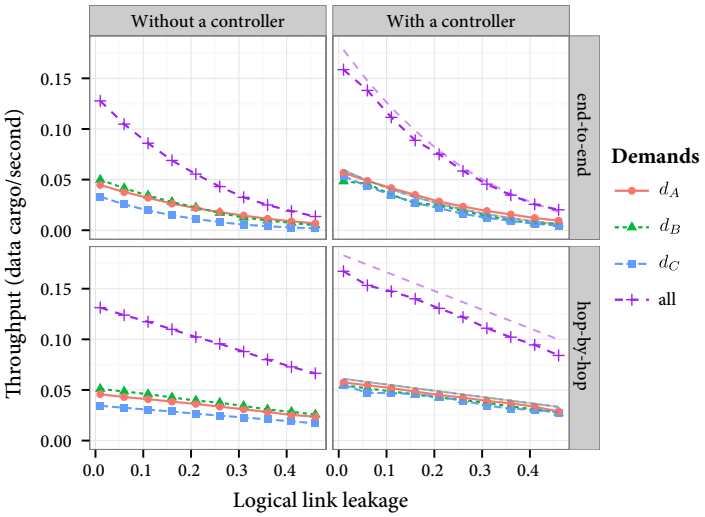


Figure 4.7. The maximum throughput achieved for offloading demands d_A, d_B and d_C (depicted in Figure 4.6) as a function of the logical link data leakage (assumed to be the same on all the logical links).

We evaluate the maximum throughput achieved by each transfer d_A, d_B, and d_C resulting from the Max-Min fairness strategy presented in Section 4.3.2. We also consider another strategy without a controller, which consists in selecting the data transfers in a round-robin order locally at each offloading spot. We consider an infinite backlog traffic generated at the single data source placed in Paris for each of the transfers (*i.e.*, $\lambda_A = \lambda_B = \lambda_C = \infty$). We evaluate the maximum throughput for each strategy and each retransmission mechanism (hop-by-hop or end-to-end introduced in Section 4.3.1).

We plot the maximum throughput in Figure 4.7 as a function of the data leakage for both the hop-by-hop and end-to-end retransmission mechanisms. The maximum throughput achieved for each of the transfers is expressed in terms of data cargo delivered per second.

Firstly, we examine the maximum throughput resulting from the strategy without a controller. We can see that this strategy does not guarantee a fair throughput distribution among the transfers resulting from the three demands. The maximum throughput for demand d_C is lower compared to the ones achieved for demands d_A and d_B. As depicted in Figure 3.5b, the data transfers resulting from demands d_A and d_C compete for the same resources, as they both follow road paths sharing common logical links. The strategy allocates the flow of vehicles traveling those links to the respective destinations of d_A and d_C without taking into account that desti-

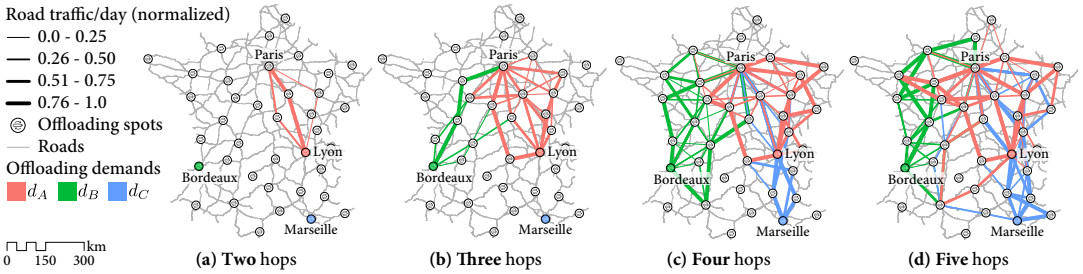


Figure 4.8. Representation of the allocation of demands d_A, d_B, and d_C resulting from the strategy with controller with different values for the maximal length of the candidate logical paths.

nation of demand d_C is farther away compared to demand d_A. Thus, the strategy favors d_A at the expense of demand d_C. The data transfer resulting from demand d_B is not affected by the unfairness of the strategy since the flow of vehicles allocated to d_B travel separate logical paths compared to demands d_A and d_C. We can also note that the resulting maximum throughput for demands d_B and d_A share the same values since destinations of both transfers are equally distant from their source.

Secondly, we examine the strategy with a controller. We can see that this strategy performs better than the strategy without a controller in terms of cumulative throughput. This result is the direct consequence of the design of the strategies. Recall from Section 4.3.2 that the strategy with controller uses the Max-Min fairness allocation model that allocates data on vehicles to maximize the overall throughput of all transfers. The higher performance of the strategy with controller further confirms its need, as already discussed in Section 4.1.

The results also confirm that the strategy with controller guarantees a fair allocation among the transfers resulting from the three offloading demands. Indeed, by design, the strategy with controller uses the Max-Min fairness allocation model that maximizes the overall throughput of all transfers while achieving a fair allocation of the flows of vehicles among all data transfers.

Finally, we can see that the hop-by-hop retransmission mechanism gives better throughput compared to end-to-end retransmission mechanism. This results from the ACKs expected from the next offloading spot as the data transfer progresses along each logical path, which results in a faster error recovery with the hop-by-hop strategy compared to the end-to-end strategy. For cargo of 1 TB in size, the allocation resulting from the strategy with a controller gives a cumulative throughput of 114 Gbps when using the hop-by-hop retransmission mechanism with a conservative 30% data leakage, which amounts to 38 Gbps per transfer on average.

4.4.2.2 Number of offloading spots involved in the data transfers

In a second step, we examine the impact of the number of offloading spots on the duration needed to complete demands d_A, d_B, and d_C. We now consider that each demand is concerned with a transfer of 10 PB of data. The flows of vehicles are allocated to each demand according to the strategy with a controller. Data losses are recovered by using the hop-by-hop strategy

given that all logical links share a data leakage of 30%. The results are shown by the bar plot in Figure 4.9. We measure the transfer duration for d_A, d_B, and d_C as a function of the maximal length of the logical paths followed by each transfer expressed in terms number of offloading spots. We also measure the mean travel time of a cargo of size $\mathcal{S} = 1$ TB. Our objective is to show the fairness of the strategy with a controller in the allocation of the transfers as a function of the degree of similarity of the paths they follow. We examine the results in Figure 4.9 together with Figure 4.8 where we represent the logical paths allocated for each demand depending on the maximal length of the candidate paths.

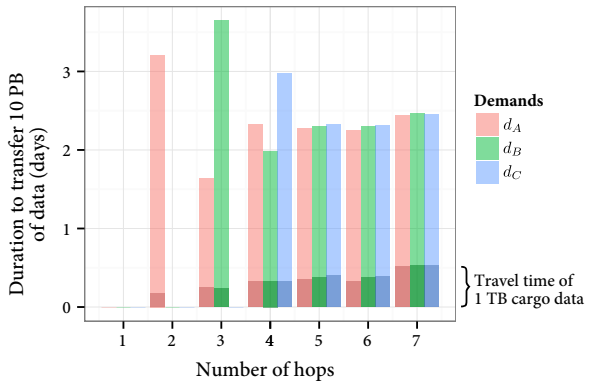
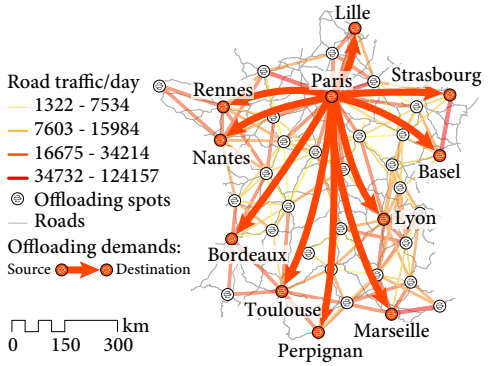


Figure 4.9. The duration needed to complete a 10 PB transfer (in lighter colors) and the travel time of a 1 TB data cargo (in darker colors) as a function of the candidate path maximal length in terms of hops.

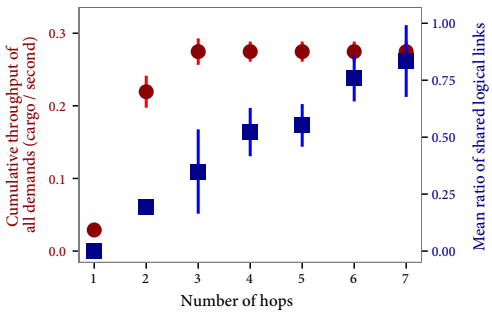
We observe that none of the three destinations can be reached with a one-hop logical path. By increasing the logical path maximal length up to two hops, Lyon becomes the only city that can be reached, as shown in Figure 4.8a. The high duration for d_A is due to the low number of paths available and therefore of allocable vehicles, which results in a low throughput. If we consider logical paths of three hops or less, Bordeaux is now reachable in addition to Lyon. Figure 4.8b shows that, in addition to the two-hop paths, there are more candidate paths between Paris and Lyon. As a result, more vehicles are allocated to d_A which decreases its transfer duration. Regarding transfer d_B, the long transfer duration is explained by the few logical three-hop paths connecting Paris to Bordeaux in a similar way to d_A and the logical paths of two-hop maximum length. With four-hop logical paths, Marseille is now also reachable, as shown in Figure 4.8c. Nevertheless, the number of four-hop logical paths is still limited between Paris and Marseille, in a similar way as the two-hop paths to Lyon and the three-hop paths to Bordeaux. What is more, Marseille is located farther away from Paris compared to Lyon and d_C competes for logical paths already passing by Lyon. This results in a longer transfer duration for d_C but also in an increase of d_A transfer duration. At the same time, increasing the length of the candidate paths to four hops enables the allocation of more logical paths between Paris and Bordeaux, which results in a clear decrease in the transfer duration of d_B. This decrease is also explained by the low degree of similarity between the logical paths allocated to d_B and those allocated to d_A and d_C, as shown in Figure 4.8c. Finally, with logical paths of five hops and more, the transfer durations are equivalent among all the demands. This further confirms that the strategy with controller guarantees a fair allocation in terms of throughput among all the demands. A slight increase in the transfer duration for all demands follows each increment in the number of hops as a direct consequence of the longer logical paths followed by all transfers. A similar trend can be observed for the travel time of the 1 TB cargo.

4.4.2.3 Cumulative capacity

We stress our system by allocating ten concurrent demands, all issued from Paris to the top nine other cities in France and one to Basel, Switzerland. This is shown in Figure 4.10a. We use the strategy with a controller to allocate the ten demands at the same time and consider again the hop-by-hop retransmission mechanism, given a 30% data leakage for all logical links. We compute the cumulative throughput achieved by all ten transfers as a function of the length, expressed as the number of offloading spots of the logical paths allocated. We also compute the mean ratio of logical links shared by the logical paths allocated to each demand over the ones allocated to all other demands. The results are shown in Figure 4.10b.



(a) Offloading demands to be allocated between Paris and the top ten other major cities of France, including Basel in Switzerland.



(b) Cumulative throughput of the allocated demands and mean ratio of shared logical links between the allocated logical paths.

Figure 4.10. Cumulative capacity with ten offloading demands spanning the French road network.

As for the previous case regarding demands d_A, d_B, and d_C, all ten destinations can be reached from Paris through logical paths of length at least three hops. With at least three-hop long logical links, the cumulative throughput of all ten demands reaches 280 Gbps and remains the same with a low standard deviation among all demands. The only city reachable by one-hop long logical paths is Lille while Lyon and Lille are the only cities reachable with paths of at two hops or less. As shown in Figure 4.10a, Lille is the closest to Paris compared to all other nine cities and can be reached by logical paths which exhibit the lowest degree of similarity with the paths connecting Paris to the other cities. For this reason, the transfer to Lille achieves a higher throughput compared to all other transfers. For the destinations located at least three hops away from Paris, the cumulative throughput remains the same even after increasing the number of hops of the allocable logical paths. Extending the length of the acceptable paths should allow the allocation of a higher number of paths to each demand and thus increase the cumulative throughput. However, most of the newly added paths are too long to be allocated or are already used for other transfers toward closer cities. For this reason, considering more paths by relaxing the length limit brings no benefit in terms of throughput. Considering longer paths to be allocable results in increased durations to complete the transfers. This further confirms the observation we made for Figure 4.9, *i.e.*, the strategy with controller allocates the paths to achieve fairness among all competing transfers.

4.5 | *Discussions*

4.5.1. On the choice of traffic counts

Annual Average Daily Traffic (AADT). The purpose of our performance evaluation is to assess the capacity enhancement brought to the Internet by the road network. Our evaluation aims at determining the amount of data that can be transferred on the road within a given time period. A typical transfer lasts from a few days to a week, as we can offload up to one Petabyte. The traffic counts we use are provided by the AADT (*Annual Average Daily Traffic (AADT)*), which are yearly averages. As so, AADTs average out the effects of seasonal and diurnal variations or missing data due to flawed monitoring. To determine the traffic volume of a road segment over the duration of a transfer, we multiply the corresponding AADT by the transfer duration measured in days. The use of the AADTs prevents our evaluation results from being affected by the diurnal, seasonal, or flawed bias.

To transpose our work in a practical setting or commercial use, thinner-grained traffic count averages should be used to account for transfers lasting few hours or several weeks. An established practice for inferring traffic count averages for different time periods consists in using temporal allocation factors applied to the annual AADTs.

An example of temporal allocation factors can be found in "The Traffic Monitoring Guide" where they are referred to as *group factors*. These factors are provided by the US Ferderal Highway Administration (FHWA) to help State Department of Transportation plan their local Highway Performance Monitoring System (HPMS) [WHYL97, Gui13]. The guide describes how to calculate the factor groups (*i.e.*, temporal allocation factors) from the traffic data collected. The procedure depends on the location and the characteristics of the road segment, as well as the time period of interest. The traffic assignment models presented in this paper remain pertinent and can be used in combination with the temporal allocation factors.

The factors resulting from this procedure are applicable for time periods of several months (seasonal), days (day-of-week), or hours. These factors determine the road traffic pattern as a ratio of the AADT for given road segments. The hourly vehicle volume $V_{i,\,t}$ on a road segment i at time t is estimated as the product of the average daily traffic AADT_i and monthly $M_{g(i),\,t}$, daily $D_{g(i),\,t}$ and hourly $H_{g(i),\,t}$ factors, each for the factor group $g(i)$ of the road segment i:

$$V_{i,\,t} = \text{AADT}_i \times M_{g(i),\,t} \times D_{g(i),\,t} \times H_{g(i),\,t}. \tag{4.8}$$

Note that the hourly factor takes into account the D-factor and the K-factor (directional and peak hour factors, respectively).

Finer-grain traffic counts. Another way to carry out the study of a dynamic system with finer-grain traffic counts (*e.g.*, in the order of the hour or for 15-minute intervals) is to characterize the dynamic flows of vehicles between offloading spots. Numerous works from transportation research have studied the estimation and prediction of dynamic origin-destination matrices using traffic counts, in particular, Bierlaire and Crittin or Casetta *et al.* both propose to do so

using sequential estimator (namely Generalized Least Squares) [BC04, CIM93]. With dynamic O-D flows between the offloading spots, one can use the recent work [FS07] that built on Ford and Fulkerson [FJF15] to study the allocation of flows over time using time-expanded graphs.

4.6 | *Conclusion*

In this chapter, we presented a centralized architecture that implements the concept of vehicular offloading. This architecture consists of a controller in charge of configuring the road path consisting in the sequence of offloading spots involved in a data transfer. To determine this road path, the controller solves a vehicle flow allocation problem formulated with a max-min fairness model. This model guarantees fair allocation among competing demands and efficient utilization of the vehicular resources. To solve the vehicle flow allocation problem, the controller computes a logical representation of the road network. The output of the allocation problem is translated to a set of forwarding rules installed at the offloading spots. Those rules indicate how much data should be loaded on the vehicles depending on their direction of travel. The controller also implements mechanisms combining retransmissions and redundancy to ensure reliable data transfers. We evaluated the fairness of the allocation procedure as well as the throughput of the data transfers offloaded on the roads of France. To calculate the capacity of the roads we used elaborated traffic modelling techniques detailed in the Appendix A which provide more realistic values. We showed that an SDN architecture provides the necessary elements to enable data offloading over the road network. With its holistic view of the offloading infrastructure, the architecture allows the management of a country-wide network, while keeping a fine-grained control of the data movements and its interactions with the vehicles to offload several Petabyte of data per day. Compared to the assessment study presented in Chapter 3, the results we presented in this chapter benefit from the use of traffic modelling techniques combined with the use of a max-min fairness allocation model.

5 Towards vehicular cloud services

In this chapter, we extend the concept of offloading spots according to two distinct directions as a basis for vehicular cloud services. In the first section of this chapter, we exploit the storage capabilities of the offloading spots to design a distributed vehicular cloud-like system to store and share file generated by mobile users. While services like Dropbox[1] and Google Drive[2] have shown their great potential given their popularity, mobile users often face limitations (*e.g.*, high cost, low bandwidth, and limited coverage) when accessing such services through wireless networks including cellular 4G LTE or Wi-Fi. In our system, the offloading spots act as repositories where mobile users upload or retrieve files. To increase the likelihood of finding the requested files in a timely fashion, the repositories distribute the user files among the other repositories by using the existing movements of the users between the repositories.

In the second section of this chapter, we dematerialize the offloading spots into pre-defined areas where vehicles come in contact frequently and long enough. We use these areas in the context of the virtualization of a large-scale vehicular network with shared vehicular resources (*e.g.*, compute, storage, and sensing) sliced into virtual machines. Virtual machines are allocated to multiple service providers, which leverage the movements of the vehicles and their shared resources to deploy large-scale geo-distributed applications (*e.g.*, sensing platforms). With these resources available, the services can benefit from a large collection of distributed mobile nodes to collect, process, and aggregate data to perform real-time analytics and make fast operational decisions [BMZA12]. We investigate the capacity of the contacts in these pre-defined areas in the context of virtual machine migrations that happen as a result of changes in the physical topology or reallocations of the vehicular resources.

In each of these two extensions, we propose an optimized placement of the offloading spots with regard to the requirements of the services. To this end, we leveraged the concepts of the logical view we introduced with the offloading overlay to represent the movements of the vehicles between the offloading spots.

Our main contributions in this chapter are:

- **Storage and sharing system** (Section 5.1). We consider a system that relies on a collection of repositories to store and share user files. We propose a placement of the repositories that exploits the movements of vehicles between the repositories to replicate files in the system.

[1] `http://dropbox.com`

[2] `http://drive.google.com`

- **Vehicle resource virtualization** (Section 5.2). We virtualize the resources of the vehicles to create large-scale virtual vehicular networks. We propose a placement of geographical areas with the required vehicle density to accommodate transfers of virtual machine migrations via vehicle-to-vehicle communications.

In the following, we present our vehicular file storage and sharing system in Section 5.1 and our vehicular network virtualization in Section 5.2.

5.1 | *Vehicular file storage and sharing system*

In this section, we present the design of a cloud-like file storage and sharing system specifically targeting mobile users in urban environments. We use the storage capabilities of the offloading spots to turn them into repositories where users store and retrieve their files. We study the impact of placement of the storage repositories such that it guarantees the availability and distribution of the files. We evaluate our placement strategy in the context of the public transportation system of buses operating in San Francisco.

5.1.1. Problem statement

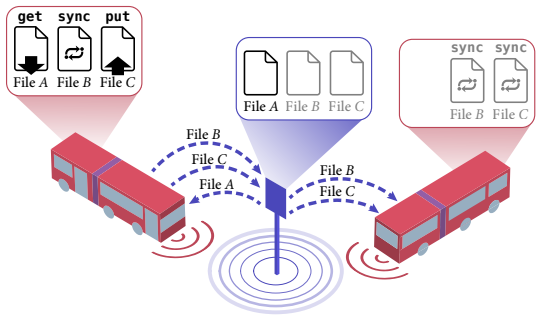


Figure 5.1. Requests and data exchanges between a repository and two mobile users when they encounter one another.

The system deploys a collection of pre-positioned offloading spots acting as file repositories. They feature high availability, unlimited storage and fast wireless transmission capabilities to handle large data rates in parallel with multiple mobile users. We assume a central controller is in charge of monitoring the files available at each repository. Whenever a repository receives a new file from a user, it notifies the controller of the availability of this file. Likewise, the mobile users can be notified of the availability of the files through the controller. We assume that the communications between the controller and the repositories and mobile users are handled by an inexpensive control channel (*e.g.*, SigFox or LoRa).

Mobile users with unlimited storage capacity upload or retrieve files to the repositories they encounter as part of their movements. There are two types of user requests: `put` requests to store user files and `get` requests to retrieve them. All requests have an associated deadline. Once a mobile user generates a request, the request becomes pending for the user until they encounter a repository. When in the transmission range of the repository, the user seamlessly transfers all of its pending requests to the repository. We assume atomic contacts between the mobile users and the repositories, which allows to transfer an unlimited number of files.

`put` request. When a mobile user generates a `put` request, the user waits to be in the transmission range of the first repository it encounters. On an encounter with a repository, the user

transfers the put request message along with the file. Once the file is received, the repository updates its local storage and notifies the availability of the file to the central controller. We define the two behaviors of mobile users when they have a put request to execute:

- "First" put policy: Mobile users immediately delete the request once they transferred it to the first repository they encounter or if its deadline expires, whichever occurs first.
- "All" put policy: Mobile users delete the request once the deadline associated with the request has expired. This allows the users to potentially replicate the files to all repositories they encounter from the moment the request is generated to the moment it expires.

get request. Users share the files they store on the repositories with other users. That is, any user can generate a get request to retrieve any available file. The user then transmits the get request to repositories it meets until the request is satisfied or until its deadline expires, whichever occurs first. In turn, the repository processes the request and transfers the requested file to the mobile user if the file is available in the repository's local storage. Otherwise, the repository notifies the mobile user that the requested file is not available. The mobile user will then try to retrieve the requested file from the next repository it will encounter. If the mobile user does not encounter any repository with a copy of the requested file before the request's deadline expires, the request fails.

The system opportunistically uses the movements of the mobile users to transport the data between the repositories and distribute the files uploaded in the system. Increased file distribution will reduce get failure rates. To accomplish this, we introduce sync requests for the files that need distribution. Repositories generate sync requests once files are stored by their initial uploader, following put requests. A sync request results in transferring copies of a file from the repository to passing mobile users in charge of distributing these copies to repositories along their trajectories. A later encountered repository updates its local storage with the corresponding file and notifies the central controller of the new availability of the file. If the file is yet to be available in every repository, the new repository starts distribution the file it just received by generating corresponding sync requests. Otherwise, the central controller informs the repositories of the distribution completion.

While replicating files to all the repositories is inefficient in terms of file storage, it fits with our assumption of atomic contacts and unlimited file storage at the users and at the repositories. We further assume no locality bias with respect to where files are generated. Additionally, we assume that we cannot predict the trajectory of the mobile users, in particular, we cannot predict the sequence of storage nodes the mobile users will visit in the future. Under these assumptions, replicating files to all the repositories is the most straightforward approach to increase the availability of files and improve the success ratio of get requests.

The three types of requests put, get, and sync are shown in Figure 5.1. In this figure, the mobile nodes are buses that aggregate requests from the passengers riding the bus. The first mobile user on the left carries three requests: get for file A, sync for file B, and put for file C. The repository, which only has a local copy of file A, executes all the requests and adds files B and C to its local storage. When a subsequent mobile user encounters the repository, the repository

generates `sync` requests and transfers them to the mobile node, along with the copies of the files. The mobile node then carries the `sync` requests to the next repository it encounters.

We consider the following metrics to evaluate the performance of our system:

- **Request success ratio.** We aim to maximize the successes and minimize the failures of user requests. A `put` or `get` request is successful if it completes before its deadline expires. The success ratio is the number of successful requests divided by the number of all requests issued during the time period of interest.
- **File distribution duration.** File distribution relies on the movements of the mobile nodes to distribute copies of files across the repositories. The distribution duration of a file is the time it takes for it to be available in all the repositories of the system. The repository placement algorithm takes this metric into account when allocating the location of the repositories to minimize the distribution duration of the files.

5.1.2. Repository placement

The goals of the placement algorithm are twofold: (*i*) determine locations such that the allocated repositories serve the maximum number of user requests before they expire and (*ii*) connect the repositories together by the mobile users' movements to create the file distribution network. The goals of the placement algorithm are represented in Figure 5.2 in the case of a bus transportation system. The bottom layer shows the trajectories of the buses in the Financial District of San Francisco, with allocated bus stops. Each bus trajectory is represented by a color and a width indicating the frequency of buses. The middle layer shows the discretized demands generated by the bus passengers. This layer shows the demands that are allocated to the repositories placed at the bus stops. Finally, the top layer shows a logical graph where nodes correspond to the allocated repositories and edges to the flows of buses running between two repositories. As in the bottom layer, the width of the edges corresponds to the frequencies of buses.

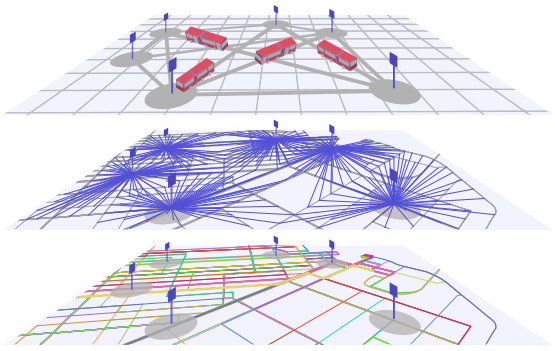


Figure 5.2. Different layers to represent the goals of the placement algorithm for a bus transportation system.

We derive the placement of the repositories from a set cover problem, in particular, the Maximal Covering Location Problem [CV74]. Given a set of candidate locations for the repositories, the placement problem consists in selecting the candidate locations that maximize the success ratio of the requests issued by the users. This problem was shown to be NP-Hard, so we adapt known heuristics to solve it [MZH83]. To this end, we consider the Greedy Adding with Substitution (GAS) heuristic that determines the optimal locations for each iteration of the algorithm [CV74].

5.1.2.1 Candidate and demand locations

The placement algorithm takes both a set $\mathcal{C}$ of candidate locations where the repositories can be placed, as well as a set $\mathcal{D}$ of demand locations to be served by the allocated repositories. The candidate and demands locations refer to geographical places (typically circles and square cells) visited by the user trajectories. The algorithm outputs a set $\mathcal{A}$ of allocated candidate locations for the repositories.

Demands. Users can generate a request at any time. Each successive location visited by the trajectories of the users is a demand location with equal probability of being the starting location of a user request. To this end, we divide the geographical area into a grid of cells of a given size (*e.g.*, $100\,\text{m} \times 100\,\text{m}$). Each cell aggregates the discretized user trajectories that go through the cell. The greater the number of trajectory aggregates, the higher the probability for a cell to be the starting location of user requests.

Candidates. Candidate locations for the repositories correspond to relevant locations the mobile users will often visit for long periods of time, allowing the transfer of significant amounts of data. For instance, candidate locations may be road intersections, or bus stops (*e.g.*, in the case of a bus transportation system). In the evaluation section, we use available data from bus transportation systems and consider bus stops as candidates for the location allocation of the repositories.

To determine the locations of the repositories, we quantify the relations between two locations l_1 and l_2 (either a demand or candidate location) by computing the following spatial statistics:

- The mean visit frequency $f(l_1,\, l_2)$ of mobile users that visited location l_1, then location l_2.
- The median inter-visit duration $v(l_1,\, l_2)$ of two consecutive visits of mobile users at location l_1 that visited location l_2 later in during their trajectory.
- The median travel time $t(l_1,\, l_2)$ of the time it takes for a mobile user to travel from location l_1 to location l_2.

We then measure the weight $w(l_1,\, l_2)$ of a pair of locations $(l_1,\, l_2)$ using the visit frequency $f(l_1,\, l_2)$ and the median inter-visit duration $v(l_1,\, l_2)$ as follows:

$$w(l_1,\, l_2) = \frac{f(l_1,\, l_2)}{v(l_1,\, l_2)}. \tag{5.1}$$

In the following, we denote by $\{w_{l_1 l_2}\}$ the *weight matrix* such that $w_{l_1 l_2} = w(l_1,\, l_2)$, and by $\{t_{l_1 l_2}\}$ the *travel time matrix* such that $t_{l_1 l_2} = t(l_1,\, l_2)$.

Relations between demands and candidates. The deadline of the user requests imposes a bound δ on the travel time $t(d,\, c)$ between a demand location d and a candidate location c. A demand location d is allocated to a candidate point c if the demand is within the bound δ and the weight w_{dc} is not null (*i.e.*, there are flows of mobile users that travel from the demand to each of the candidates). The goal is to maximize the sum of the weights of the demands

$d \in \mathcal{D}$ allocated to a candidate c: $\sum_{d \in \mathcal{D}} \{w_{dc} \mid t_{dc} \leq \delta_{dc}\}$. Note that a single demand cell can be allocated to multiple candidates, each within the given bound δ with non-null weights.

Relations between pairs of candidates. We characterize the connectivity between two candidates in terms of flows of mobile users travelling from c_1 and c_2 with both weights $w_{c_1 c_2}$ and $w_{c_2 c_1}$. For each candidate, we maximize the sum of the weights between this candidate c and the chosen candidates $k \in \mathcal{A}$: $\sum_{k \in \mathcal{A}} w_{kc} + w_{ck}$.

5.1.2.2 Location allocation algorithm

The algorithm derives from greedy-adding with substitution (GAS) heuristic proposed to solve the Maximal Covering Location problem [CV74]. For each candidate location, we divide the iterations into two parts: selection and substitution.

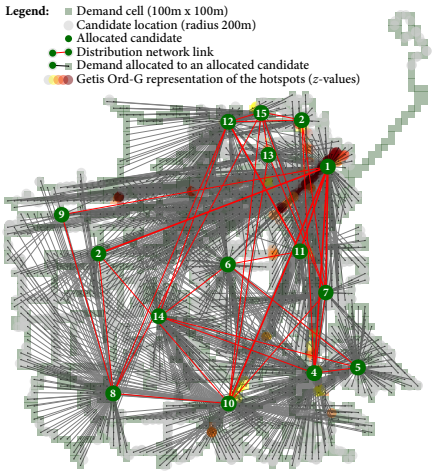
Selection. We first pre-select the set of candidates from the set of remaining candidates such that each candidate is connected to the chosen repositories by the flows of mobile users. This pre-selection is necessary, as it guarantees the creation of the distribution network and increases the availability of files. From these pre-selected candidates, we choose the candidate that maximizes the sum of demand weights within the bound δ as we defined previously.

Substitution. The second part of the algorithm goes back on the previous iterations and examines each allocated candidate and tries to find a better candidate that maximizes the demand weights, while guaranteeing that the new candidate will be connected to the chosen candidates. If a better candidate is found, it replaces the candidate under examination. The substitution part allows to replace the candidates allocated in the early iterations that are not justified in the later iterations due to subsequent allocated candidates.

We adapt the GAS heuristic to our system as follows. The first candidate is chosen such that it maximizes the sum of demand weights within the bound δ. Note that, since the demand weights are maximized for this candidate, the candidate is often visited by mobile users, making it a good intermediary node to be connected to the other candidates by the flows of mobile users. There is no substitution part, as no other candidate was already chosen. In our implementation, once a candidate is chosen, we remove all the candidates in the vicinity of the chosen candidate (in terms of travel time and distance from the chosen candidate, *e.g.*, half the deadline and the distance corresponding to this duration with average speed). This allows us to better spread the repositories over the geographic area and avoids cluttering denser regions with too many repositories.

5.1.3. Real experiment simulation

We run our evaluation in two parts. The first part corresponds to the allocation of a given number of repositories for the given deadline. The second part is the simulation of the system with the repositories placed in the area.



(a) Allocation of the repositories. The color variations of the demand and candidate locations represent hotspots resulting from the Getis Ord-G spatial analysis with respect to the bus visit frequencies at these locations [GO92].

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	51 (7)	9 (3)	28 (5)	35 (8)	54 (12)	52 (9)	31 (6)	40 (5)	87 (13)	49 (5)	34 (7)	6 (3)	9 (3)	47 (8)	0 (0)
14	55 (7)	51 (8)	63 (9)	32 (12)	36 (10)	69 (12)	40 (12)	63 (12)	108 (17)	40 (12)	48 (9)	41 (9)	57 (9)	0 (0)	49 (10)
13	50 (6)	21 (5)	39 (6)	42 (5)	58 (10)	50 (7)	35 (8)	52 (6)	96 (13)	44 (6)	31 (5)	18 (5)	0 (0)	59 (9)	10 (3)
12	46 (5)	15 (4)	23 (5)	38 (6)	56 (12)	46 (9)	37 (5)	35 (4)	81 (14)	54 (4)	27 (7)	0 (0)	16 (5)	41 (10)	7 (3)
11	19 (3)	40 (6)	49 (5)	17 (4)	31 (9)	19 (6)	25 (6)	62 (6)	84 (18)	48 (7)	0 (0)	27 (6)	33 (7)	47 (10)	33 (6)
10	31 (4)	52 (12)	40 (8)	45 (15)	59 (19)	61 (6)	30 (17)	24 (8)	69 (12)	0 (0)	47 (7)	54 (4)	45 (6)	41 (13)	50 (5)
9	70 (20)	95 (11)	61 (13)	92 (19)	108 (19)	96 (20)	83 (16)	44 (14)	0 (0)	71 (14)	84 (17)	84 (14)	98 (14)	108 (17)	90 (13)
8	52 (6)	50 (7)	14 (4)	64 (10)	80 (15)	78 (7)	53 (13)	0 (0)	46 (14)	25 (8)	59 (5)	34 (4)	50 (6)	63 (12)	41 (6)
7	28 (12)	23 (8)	54 (6)	9 (7)	27 (11)	44 (7)	0 (0)	59 (14)	89 (18)	32 (17)	26 (6)	38 (6)	38 (9)	39 (12)	35 (9)
6	33 (4)	61 (8)	67 (6)	36 (7)	39 (8)	0 (0)	43 (7)	80 (6)	99 (19)	62 (6)	19 (6)	47 (8)	53 (10)	67 (12)	55 (8)
5	44 (12)	43 (11)	72 (9)	12 (9)	0 (0)	41 (12)	23 (10)	75 (12)	104 (20)	50 (14)	32 (11)	54 (10)	54 (9)	36 (12)	52 (8)
4	28 (6)	28 (9)	59 (6)	0 (0)	14 (8)	36 (7)	9 (6)	65 (12)	94 (19)	40 (15)	16 (4)	40 (6)	42 (6)	30 (12)	39 (6)
3	41 (8)	39 (6)	0 (0)	59 (6)	79 (12)	67 (7)	55 (6)	16 (3)	64 (16)	41 (8)	49 (6)	23 (4)	39 (6)	63 (8)	30 (6)
2	50 (11)	0 (0)	40 (6)	26 (9)	45 (12)	58 (8)	24 (9)	53 (6)	96 (14)	51 (12)	40 (6)	17 (5)	21 (5)	50 (7)	12 (4)
1	0 (0)	49 (11)	40 (8)	27 (5)	44 (11)	32 (5)	28 (12)	52 (6)	68 (20)	31 (4)	17 (3)	45 (8)	50 (8)	60 (12)	50 (7)

(b) Origin-destination matrix of the repositories (identified by the numbers represented on the Figure 5.3a showing the mean travel time in minutes between each pair of repositories. The values in parentheses are the corresponding standard deviation. The simulations were done for 15 repositories placed as shown in Figure 5.3a.

Figure 5.3. Results for the allocation of 15 repositories in San Francisco using the bus movements of the MUNI bus transportation system.

Mobility model for public transit vehicles. Firstly, we simulated the movements of the San Francisco MUNI buses[3] using the ONE simulator [KOK09]. The ONE simulator is a time-discrete and event-based simulator of delay-tolerant networks. We used the `MapRouteMovement` mobility model implemented in ONE to simulate the movements of mobile nodes and their stops on a street map. This mobility model is particularly well suited to simulate the movement of buses that are part of a transit system. We derive the movements of the buses from GTFS (*General Transit Feed Specification*)[4] feeds given by the MUNI transit authority that gather the buses' schedules. We develop generic tools to convert the GTFS feeds into a collection of `MapRouteMovement` mobility models compatible with the ONE format.

Repository placement. Secondly, we developed a framework to determine the locations of the repositories. The framework implements the algorithm described in Section **??**. It takes the bus movements we derived from the GTFS feeds. We divide the geographical area into a grid of cells of size 100m $\times$ 100m. We take bus stops as candidates to host a repository, since the buses stop for a long enough duration to transfer large amounts of data. The framework then allocates the locations of the target number of repositories among the candidate locations, given user request deadline. We represented in Figure 5.3a the output allocation of 15 repositories' locations out of the 4,590 bus stops in San Francisco using the movements of the MUNI buses. In this figure, we show a Getis Ord-G hotspot analysis of the spatially significant bus stop candidates with regards to the visit frequencies of the buses in their vicinity [GO92].

[3] `https://www.sfmta.com/`

[4] `https://developers.google.com/transit/gtfs`

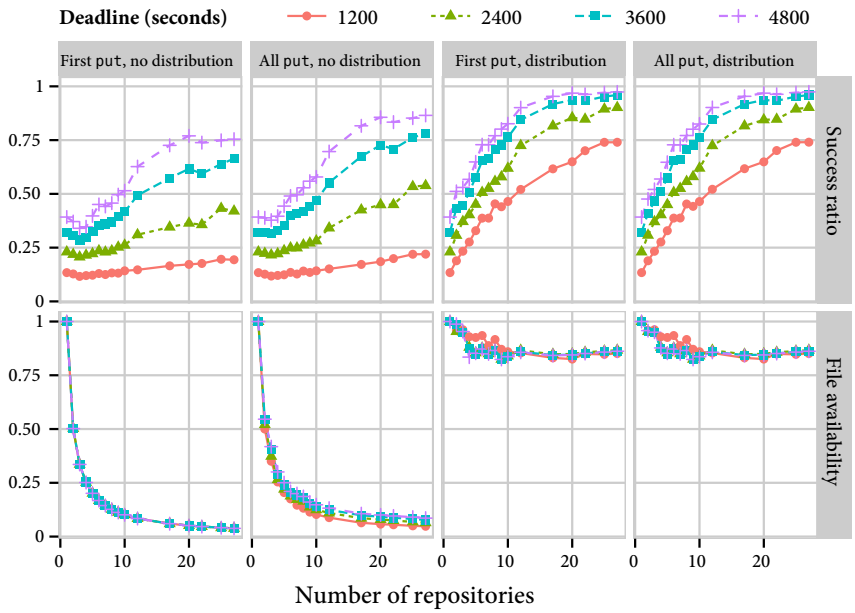


Figure 5.4. Comparison of the impact of the different `put` policies and the existence of the distribution network in the user request success ratio and the file availability in the repositories.

After running our methodology to convert the GTFS feed, we obtained the movements of 493 buses running on 130 different trajectories, which represents the traffic of buses running between 10am and 4pm on a typical weekday. Both the mobile users and the repositories are equipped with WiFi network interfaces with a range of 250 m. We ignore wireless interference and assume enough wireless capacity to accommodate the full exchange of files between the repositories and the mobile users. The candidate locations are the bus stops given by the GTFS bus feed projected onto the street map. The wait time of the buses at the bus stops is picked at random between 10 seconds and 30 seconds. The mobile users send requests independently every 10 minutes following a Poisson distribution, without any assumptions on the popularity and geographical locality of the files. We assume the mobile users generate the `put` and `get` requests as follows: 10% of the requests are `put` requests and 90% of the requests are `get` requests. We assume the mobile users generate the `put` and `get` requests. The simulation duration is 20,000 seconds. For each simulation, we deploy a target number of repositories up until the optimal number of repositories, where no demands are left unallocated.

Success ratio. As a first step, we perform the simulations to evaluate the benefits of the `put` policies and the distribution network in the success ratio of the user requests. To this end, we simulate the system under different assumptions regarding the `put` policy ("first" or "all") and the existence of the distribution network. We represent the average success ratio of the `get` user requests and the file availability in the repositories in Figure 5.4. In the scenario with the "first" `put` policy and no distribution network, we notice that the more repositories there are, the more user requests are satisfied. Some buses that have pending `get` requests are not able to reach any repository because of their routes and the few repositories available. Adding

more repositories allows more buses to reach repositories, resulting in better success ratio of the `get` user requests. However, since the data is scattered across multiple repositories, the success ratio of the `get` requests still remains low for short deadlines. For long deadlines, the buses visit more repositories, which increases their chances to visit one with a copy of the requested file. Taking advantage of the "all" `put` policy helps to increase the success ratio of the `get` requests by 9.98% on average for all the deadlines. With this policy, the buses distribute copies of the files to different repositories, thus increasing the availability of the files. Contrary to the "first" `put` policy where user files are available at only one repository, the "all" `put` policy increases the chance of a mobile user to find a repository with a local copy of the requested file.

The distribution network led to increase the user request success ratio by 104.3%. Note that this improvement is greater for lower user request deadlines (*e.g.*, the improvement for the user requests of 1200 seconds is of 201.4% on average). The distribution network distributes copies of the files in several repositories. Since the repository placement algorithm guarantees that all the repositories are connected together, the files can be distributed to each repository of the system. Hence, a mobile user with a pending `get` request is more likely to encounter a repository that has a local copy of the requested file, which increases the success ratio of the `get` requests. Note that the use of the "all" `put` policy does not change anything in the request success ratio and file availability compared to the "first" `put` policy, as the mobile user that originated the request is also part of the distribution network.

File availability. We compare the benefits of the distribution network with the "all" `put` policy using the "File availability" plots in Figure 5.4. The plots show the proportion of files available in the repositories, where "1.0" of file availability corresponds to all the files being available at each repository. We clearly see the benefits of the distribution network to distribute the files among the repositories. The "all" `put` policy also distributes the files to several repositories, however, it fails to distribute the files to repositories beyond those located on the routes of the mobile users that generated the `put` requests. The plot further shows that, even with the distribution network, the files are not available at every repository (only 85% of them). In fact, the data represented accounts for the files that were not fully distributed by the end of the simulation.

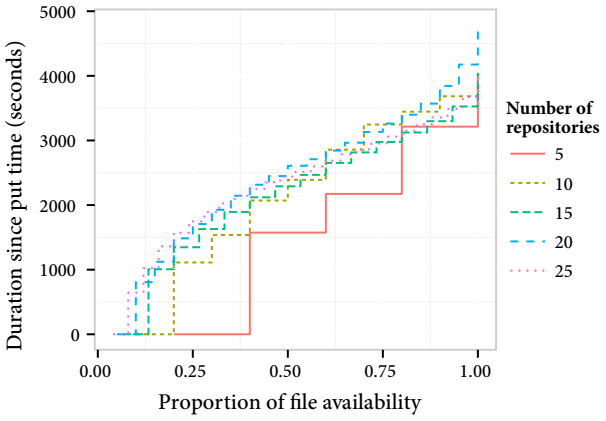


Figure 5.5. Distribution duration as a function of the file availability for different numbers of repositories.

Distribution duration. In the second step of our evaluations, we study the distribution duration of the copies of user files among the available repositories. In Figure 5.5, we show the average duration to distribute a file throughout the distribution network since the time the file was stored in the first repository (following a `put` request). We represent the distribution duration of the copies of user files as a function of the proportion of the file availability for different

numbers of repositories. For example, with 10 repositories, 40% of file availability corresponds to the file being available in four repositories. The first repository where the file is stored can be any of the repositories available. We notice that the average time needed to distribute the user files to every repository, regardless of the number of repositories available, is 4000 seconds, or a little more than one hour. It takes more time to distribute the copies of the files to the repositories at the beginning and end of the file distribution. At the beginning of the file distribution, only one copy of the files is available in the first repository. It takes on average 700 seconds to distribute the first copy of the file to the second repository. As more repositories distribute copies of the files, the distribution of the copies becomes faster. By the end of the file distribution, copies of files are available in most repositories. It takes on average 500 seconds to reach the last repository, as it is the farthest away from the first repository where the original copy of the file was stored.

In Figure 5.3b, we give an origin-destination matrix that shows the average travel time in minutes between any pair of repositories. This translates to the time needed to distribute the file to different repositories from the time the file was stored in a first repository. These values give the average duration for a file to be available at a repository, depending on the originated repository. In the figure, the repositories are identified by the same numbers as the ones in Figure 5.3a. The connectivity between two repositories depends on the number and frequency of the buses going from one repository to another. For instance, repositories 1 and 15 are very well connected to the rest of the repositories. However, repository 9 is not as well connected since it is farther away from the rest of the repositories. This further explains the longer time it takes to distribute copies of the user files from repository 9 to every other repository. This goes to show that the further away the repositories are, the more time it takes on average to reach them and distribute copies of the files to them.

5.2 | *Virtual machine migration in vehicular networks*

In this section, we dematerialize the offloading spots into areas where the vehicles often come in long contacts via Vehicle-to-Vehicle (V2V) communications. We use these areas in the context of large-scale mobile networks to address the problem of virtualizing the resources of vehicular nodes (*i.e.*, compute, storage, and sensing) to create a virtual vehicular network. The vehicular resources are managed by an infrastructure provider through a centralized controller and allocated to service providers who configure the virtual machines to deploy geo-distributed services (*e.g.*, large-scale sensing platform). In particular, as shown in Figure 5.6, we focus on the migration of virtual machines triggered by the reallocations of virtual resources or changes in the physical topology. Instead of using cellular connectivity, we leverage vehicle-to-vehicle communications to migrate the virtual machines.

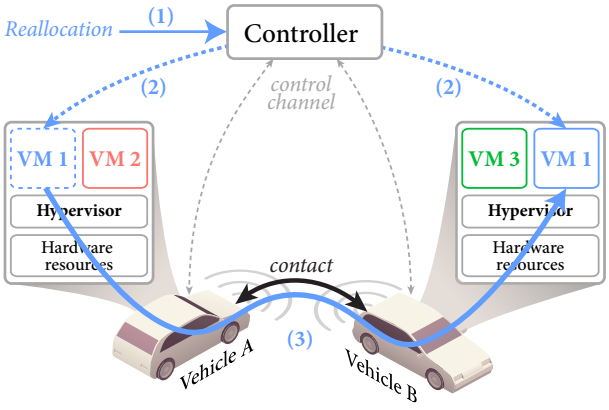


Figure 5.6. Virtual machine migration between two vehicles in contact: **(1)** The controller receives a reallocation demand of the resources on vehicles A and B: VM 1 has to move from A to B. **(2)** The controller notifies the two vehicles to initiate the migration during their next contact. **(3)** When there is a contact, the VM is transferred from A to B.

This contribution draws on the results presented for various cities including San Francisco and Warsaw where it was shown that vehicles come in contact often and for long period of time in specific areas [KWSB04, SDPG06, UTCFBO14, NF13]. We present a generic methodology for analyzing contact opportunities in urban scenarios where mobility traces are available for various types of vehicles. By limiting the virtual machine migrations to these specific areas, the expected benefit is an improved ratio of successful migrations. We first present this methodology and then evaluate its benefits in the context of virtual machine migrations among the public buses of the city of Dublin, Ireland.

5.2.1. Contact density heatmap generation

The methodology consists first in processing the mobility traces with the purpose of generating a heatmap representing graphically the spatial variation of the contact density. The next steps consist in determining the location of the hotspots matching specific values regarding the duration and the rate of the contacts occurring in those areas. The different steps of the methodology are depicted in Figure 5.7.

Contact heatmap. The first step consists in computing the spatial variation of the contact density. The geographical area of interest is divided in a grid pattern where a square cell has a fixed size, *e.g.*, 100 m $\times$ 100 m. In each cell, the only contacts plotted on the heatmap are the relevant ones which in the VM migration scenario are the contacts lasting for at least 200 s. This

duration amounts for the time needed to transfer a typical virtual machine of 200 MB. A kernel density estimator is then used to characterize the contact density of each cell. This consists in clustering the points representing the contacts occurring in the cell. Larger numbers of clustered points result in larger density values [Sil86]. The heatmap obtained for the city of Dublin is shown in Figure 5.7b where different contact densities are depicted using different shades of red, the darker shade indicating a higher density of contacts. We notice that the contacts between buses are concentrated in the city center, but also on the outskirts of the city center, along the major traffic arteries and at some of the main intersections of Dublin. The next steps consist in deriving the hotspot locations from the contact density heatmap.

Contact hotspots. Firstly, the cells are ranked according to their contact density. To exclude cells where contacts are not occurring in significant numbers, the 25% top cells are selected. The contiguous cells are then aggregated in clusters referred to as *hotspots*. Hotspots are represented on the map by circles centered so they can cover the corresponding cluster. The diameter of the hotspots is fixed and selected so the largest cluster is fully covered by a single hotspot.

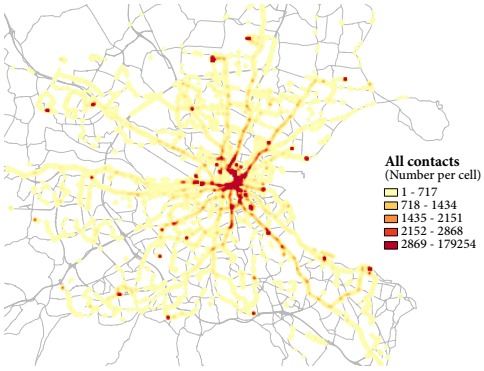
The hotspots depicted in Figure 5.7b are further characterized from the perspective of the V2V migration of virtual machines. For each hotspot, the two following variables are computed: The total number of contacts with a duration of at least 200 s (n) and the average inter-contact duration (x). The inter-contact duration refers to the duration between two consecutive contacts occurring within the same hotspot. Those two variables are then used to assign a score to each hotspot we compute as: $n/\max(1, x)$. Finally, we use the Jenks optimization method to identify three value ranges which provide the best arrangement of the hotspots according to their scores [Jen63].

In Figure 5.7b, different shades of color are used to represent the hotspots according to their classes. In the performance evaluation, two significant hotspots are selected, each belonging to the two higher classes: "Hotspot 1" has the highest score which indicates that long-duration contacts occur with high frequency. Note that three other hotspots share a similar score. The second hotspot indicated as "hotspot 2" is selected arbitrarily among the 16 occurrences sharing a similar score.

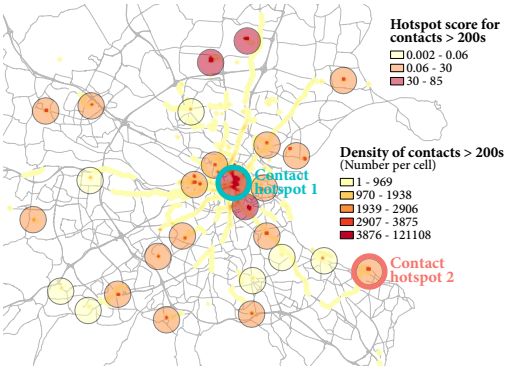
5.2.2. Experimentation with a bus system

In this section, we assess the feasibility of migrating virtual machines in the context of the public transportation system of the city of Dublin. In our simulations, we focus on the capacity of the opportunistic contacts between buses while in transit.

Experimental setup. For our feasibility study, we use a publicly available dataset of real traces of Dublin city buses provided by Dublinked, part of the Dublin City Council. The mobility traces span the month of January 2013 and provide timestamped GPS coordinates of all Dublin-Bus buses in service [Dub13]. We analyze the movements of 823 buses during the service day of Tuesday, January 29th, 2013 from 10am to 1pm. We assume that two buses are in contact whenever the buses are in each other's communication radius denoted R. We used a recent



(a) Heatmap of the contacts of any duration in Dublin, Ireland.



(b) Heatmap of contacts whose duration is greater than 200 seconds and the hotspots centered at location centered at the aggregated 25 % densest cells.

Figure 5.7. Heatmaps of contacts between buses in Dublin, Ireland. The figures denote the number of contacts per cell (of size 100 m $\times$ 100 m). We choose to simulate migrations between buses entering the two hotspots circled on the map of Figure 5.7b.

map-matching algorithm proposed by Newson and Krumm to match the sequence of timestamps positions with the planned route given in the GTFS feed provided by the Dublin City council [NK09]. This step allowed us to remove outliers from the simulation, which could bias the results, as well as form more realistic trajectories by adding timestamps waypoints corresponding to the respective routes followed by the buses.

We use the ONE, a discrete-event simulator to run our experiments [KOK09]. The ONE can use real-world mobility traces to reproduce the movements of DTN nodes using linear interpolation between two timestamped waypoints and infer their contacts. We used the default settings of the ONE to simulate IEEE 802.11 on the mobile nodes. However, the only layers supported by the ONE simulator are the network layers and above. To account for the properties of the physical layer including the link-level packet losses, we select appropriate values regarding the node communication radius and contact transfers goodput. We draw on the results presented by Rubinstein *et al.* which show a 1 Mbps plateau in the TCP goodput when the contact distance between vehicles is less than 150 m [RBADdA$^+$09]. The choice of a conservative node radius is further supported by Hadaller *et al.* who showed that the Packet Delivery Ratio (PDR) measured at the MAC layer remain stable between vehicles in contact within a similar distance of 150 m [HKBA07]. In our simulations, we considered a conservative communication radius R of 150 m with a homogeneous average goodput of 1 MB per second.

VM migration simulations. In the remainder of this section, we simulate the migration of virtual machines of various sizes running on every bus. A bus entering a specific hotspot initiates the migration of a virtual machine with the first bus it comes in contact with, following the First Contact routing policy implemented by the ONE simulator. We make no assumption regarding the capacity of the buses to predict the duration of a contact. A transfer will be aborted in case the contact duration is not long enough to accommodate the transfer. Hence, multiple

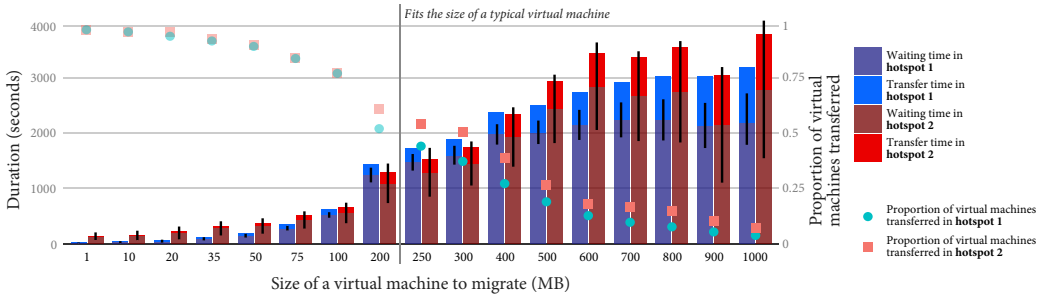


Figure 5.8. Proportion of successful virtual machine migrations and mean time to transfer the virtual machine as a function of the virtual machine size. The black solid bars represent the 95% confidence interval for the waiting time before successfully transferring the virtual machines.

attempts may be needed before the migration succeeds. The migration is considered as failed if the bus exits the hotspot without succeeding in transferring the virtual machine.

The results we present reflect averages over 100 virtual machine migration trials for different sizes of virtual machines we try to migrate at each of the two hotspots introduced in Figure 5.7b. The range of values we use captures the usual sizes of virtual machines that can be found, that is, from a few hundreds of kilobytes (*e.g.*, TinyOS [LMP+05]) to a few hundreds of Megabytes [CFH+05]. For each size of virtual machine, we measure the average time needed for a bus entering each hotspot to find a suitable contact and the time it actually takes to migrate the virtual machine. The plot in Figure 5.8 shows the mean time to transfer virtual machines of different sizes, as well as the proportion of virtual machines that were actually transferred.

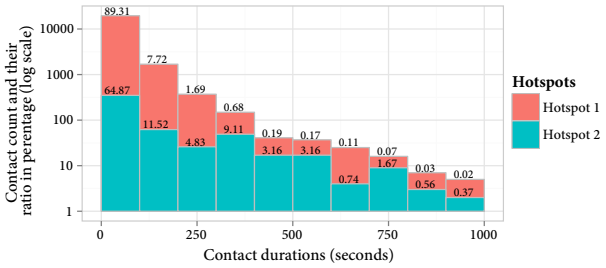


Figure 5.9. Number of contacts (in log scale) accounted for different duration ranges and their ratios (in percentage) against the total number of contacts in the hotspots 1 and 2, respectively.

First, we note that the ratio of virtual machines successfully migrated at both hotspots decreases as their size increases. This is due to the lack of enough long-lasting contacts between buses. Buses fail several times before finding a suitable contact to transfer the virtual machine. This observation is confirmed by the time spent waiting for a contact suitable in duration with large-sized virtual machines. Overall, it takes less time to find a suitable contact at hotspot 1 than at hotspot 2. Buses entering hotspot 1 spend less time waiting for a suitable contact in comparison with hotspot 2 expect for virtual machine sizes ranging from 200 to 400 MB. To further explain this trend, we plot in Figure 5.9 the total number of contacts measured for different contact durations as well as the ratios of those contacts against the total number of contacts occurring in hotspots 1 and 2. We can see that hotspot 2 has a higher ratio of contacts lasting long enough (*e.g.*, 200 s and 400 s) to accommodate transfers of virtual machines with a size ranging from 200 MB to 400 MB. Despite the higher number of contacts at hotspot 1, most of them are not suitable for the transfers of such virtual machines. Buses spent more time trying transfers that will eventually fail before

finding a suitable contact. This also explains why the ratio of successful migrations is higher at hotspot 2 when virtual machines have sizes ranging from 200 to 400 MB.

5.3 | *Discussions*

Atomic contacts (first extension). The stronger assumption we made is to consider contacts between mobile users and repositories with unlimited capacity. If we want to relax this assumption, an increasing number of requests would not be satisfied with the simulation time. We need to consider the following enablers. Firstly, with equal file size and control messages with negligible size, we have to adapt the placement algorithm to take the contact duration of buses at each candidate repository. In particular, we need to differentiate the (longer) contacts of buses with repositories located in bus stops where the buses will actually stop at, and the (shorter) contacts between buses and repositories located in bus stops where the bus does not stop at. Secondly, we need to prioritize the requests to transfer between the repository and the bus. As an example, the `put` requests must be transmitted to the repository before any other request. Similarly, if the repository can satisfy the `get` requests carried by the bus, the corresponding files must be transmitted before the files associated with the `sync` requests of the bus and the repository. Finally, we note the different priorities impacts the replication of the data across the repositories. As a result, we need to develop new strategies to decrease the number of copies to replicate.

Replication strategies (first extension). With the replication of file copies to every repository, a large number of copies is distributed and received by the repositories, as we show in Figure 5.10. With non-atomic contacts, we need to limit the overhead of copies sent over contacts between mobile users and repositories. With no knowledge of the future trajectories of mobile users, one could distribute copies of files for a bounded duration until the distribution of copies of the file stops. Alternatively, one could bound the number of copies generated per repository and transmitted to the mobiles users. With techniques borrowed from Delay Tolerant Networks, one could use a control plane that would enable the repositories to "learn" the frequent repositories the mobile users encounter along their route [LDS03, GV06]. With this information, a single copy would be needed to transmit per repository.

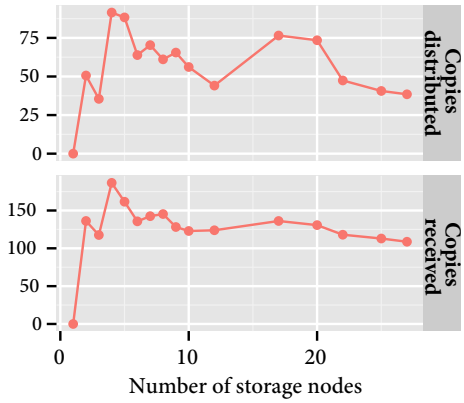


Figure 5.10. Average number of copies distributed and received by the repositories for a file and a user request deadline of 2400 seconds.

Virtual machine allocation (second extension). Recall that we considered a set of virtual machines managed by an infrastructure provider who allocates them to service providers who deploy their services. However, we did not detail the algorithm to allocate the virtual machines to the service providers such that the allocation result satisfies their requirements. As an example, the requirements may relate to the number of virtual machines to allocate, a target guaranteed

connectivity of virtual machines through vehicle-to-vehicle connectivity, or the geographic areas covered by the virtual machines. The output of the allocation algorithm must minimize the number of migrations and where they happen in order to minimize the duration of the transition. Additionally, the service providers may issue new demands to the infrastructure provider or change their current demands. To this end, the latter must determine whether the demands can be satisfied before serving them. An approach to achieve this is to check the current state of the available and allocated resources through the control channel before allocating new resources. If the demand can be allocated or modified, the infrastructure provider re-runs an allocation algorithm that triggers new slicing of the resources of the nodes, as well as virtual machine migration between the mobile nodes.

Finding suitable contacts (second extension). In order to successfully transfer a virtual machine between two vehicles, the contact must be long enough to accommodate the size of the virtual machine. In our evaluation, we proposed a methodology to determine the geographical areas we refer to as hotspots where long-duration contacts occur at significant rates. Hotspots are located in geographical areas within which all contacts amount for 75% of all 200 second long contacts and occur at most every 1000 seconds. However, although long-duration contacts occur statistically more often in the hotspots compared to other areas, there is still a high density of short-duration contacts (as shown in Figure 5.7b). Depending on the first contact a bus has when entering a hotspot, the VM migration may fail if the first contact is not long enough to accommodate the target virtual machine size. One way of improving the migration success is tu use models to predict contact durations between buses. Many existing techniques may be borrowed from the DTN literature. For instance, this could be done by maintaining a list of contact with their corresponding durations for each vehicle encountered [LDS03, GV06] or by exchanging the direction and speed or itinerary between vehicles in contact to estimate the shared route with the other vehicle [CLGH10, BCL07].

5.4 | *Conclusion*

In this chapter, we proposed to extend the concept of offloading spots into two distinct directions in the context of different vehicular cloud services. In both extensions, we relied on a logical map to characterize the movements of the vehicles within the offloading spots or between them. We then used this logical map to determine the placement of the offloading spot to optimize the resulting compositions of the movements of the vehicles according to the service requirements.

In the first extension, we proposed a placement of offloading spots that act as repositories that are part of a distributed cloud-like storage and sharing system. The placement aims to satisfy a maximum of user requests to store or retrieve files and distribute the files throughout the repository network using the movements of the mobile users between the repositories. We showed the benefits of using the movements of vehicles to distribute files and thus improve the success ratio of the user requests.

In the second extension, we dematerialized offloading spots into pre-defined areas where vehicle often meet long enough to transfer large amounts of data. We used these areas in the context of a virtualization of the resources of the vehicles to migrate virtual machines following topology changes. We proposed a placement of these areas such that the vehicles can accommodate large data transfers of the size of a virtual machine. With simulations of real-world bus traces, we showed that these areas increase the success of the transfers as the vehicles are more likely to come in contact with other vehicles for long durations. As a result, transfers of several hundreds of Megabytes can be achieved through V2V communications, enough to accommodate migrations of virtual machines.

These extensions showed that the existing mobility of vehicles has the potential to provide innovative added-value services for urban environments, with limited reliance on conventional data networks.

6 Conclusion and perspectives

6.1 | *Summary of contributions and takeaways*

The work presented in this thesis exploits the existing mobility of everyday entities, including private vehicles and bus transportation systems to overcome the limitations of conventional data networks. We used the mobility in the context of various applications, including traffic offloading and cloud-based services.

Massive data offloading. We equipped vehicles with storage capacities connecting specific locations we referred to as offloading spots. The offloading spots avoid relying on the vehicles that make a direct trip from the source to the destination of the data transfer. The offloading spots help maximize the utilization of the combined storage of vehicles traveling the road segments connecting the offloading spots. To this end, they act as data relays where the trajectories of the vehicles making stops are composed into a single path followed by the data they carry. In the following, we detail how we used the offloading spots to control and manage the composition of the movements of the vehicles to match the requirements of the data transfers.

Control and management of the offloading infrastructure. Offloading spots are also equipped with storage where data is buffered and asynchronously passed from one vehicle to another. The offloading spots are located where vehicles stop long enough to transfer large amounts of data. Examples of such locations include parking lots, shopping centers, and gas stations. We proposed an SDN architecture with a centralized controller that has a holistic representation of the offloading infrastructure. The controller manages the resources of the offloading system using a logical representation of the dynamics of the road network. This representation is called the offloading overlay and results from a mapping algorithm that translates the movements of vehicles in terms of network quantities. The offloading overlay consists of logical links, each characterized by its capacity, delay, and loss rate. The controller leverages the offloading overlay to solve the vehicle flow allocation problem, which consists in selecting and allocating data to the vehicle flows connecting the offloading spots. The resolution of the allocation problem maximizes the utilization of the road resources. The output of the allocation is translated into a set of rules the controller uses to configure the offloading spots. These rules indicate which vehicles an offloading spot should select and which data to load onto the vehicles. Our results on the main roads of France with real traffic counts showed that the centrally controlled architecture can achieve an efficient and fair allocation of concurrent data transfers between major cities in France.

Vehicular storage and sharing system. We exploited the storage of the offloading spots to turn them into repositories where mobile users can store and retrieve files. The system relies on a collection of repositories and the movements of the vehicles between them to distribute the files among the repositories. We proposed a placement algorithm to determine the optimal locations of the repositories so each repository can satisfy the user requests to store or retrieve a file. The placement of the repositories also considers the movements of the vehicles connecting each pair of repositories so the files uploaded to the system can be synchronized without relying on the existence of a data network. We showed that the distribution of files using the existing mobility of the vehicles helps increase the success rate of user requests.

Virtual vehicular network. We dematerialized the offloading spots in the context of a vehicular network where the vehicle resources, including storage, but also processing or sensing are virtualized with virtual machines. We studied the virtual machine migration through vehicle-to-vehicle communications. In this context, the offloading spots are areas where vehicles come in contact often and for long periods compared to other areas. We showed that restricting the virtual machine migrations to these areas improves the success rate of the migrations.

Main challenges. During this thesis, I had to find data sources through people I contacted in different institutions and companies. While most of these attempts were unsuccessful, I managed to find relevant local data following recent open-data initiatives, as an increasing number of institutions provide publicly available data. Once I obtained the data, I had to find ways to analyze it in order to characterize it into network quantities. Unfortunately, most of the open-source traffic simulators (SUMO [BBEK11] or MatSim[1]) were unsuited for the analysis of the road networks. I had to develop my own set of tools to perform the analysis and characterization of the networks necessary for the evaluations. Finally, I had to study and understand the relevant body of work borrowed from transportation research to infer flows of vehicles from traffic counts and characterize them into network quantities.

To summarize, the key takeaways from this thesis are:

- The vehicles in circulation are an untapped potential for data transportation. We illustrated this assertion with simulations of vehicles transporting data on French roads and showed that the road network has the potential to daily offload several Petabytes of data.
- A dedicated infrastructure increases the overall capacity of the system by composing the trajectories of the vehicles and configuring the path followed by the data.
- A centralized control of the infrastructure with a holistic representation of the vehicle flows enables efficient management and allocation of the road resources to data transfers.
- A logical representation of the vehicular resources into network quantities mitigates the scale of the road network and makes the data transfer allocation tractable.
- The mobility of everyday entities gives the potential to new actors operating mobile entities (*e.g.*, vehicle owners or bus transit agencies) to provide value-added services with limited reliance on conventional data networks. We illustrated this assertion with our offloading system and the extensions that create vehicular cloud services.

[1] `http://www.matsim.org/`

6.2 | *Perspectives*

6.2.1. Short-term perspectives

Offloading spot placement. In our data offloading, we created a realistic deployment of charging stations. We emulated charging demands issued within cities we weight according to their sizes. The charging stations are placed so as to satisfy these demands. As a result, each charging station is located within the range of the surrounding electric vehicles. One could consider the charging demands made by vehicles while on the move. Some work considered charging demands issues by the traffic flows themselves [Hod90]. These deployment strategies make sense in the context of charging station networks such as the one consisting of the superchargers operated by Tesla. Instead of considering the charging demands, one could propose an alternative deployment strategy derived from the requirements of the data offloading. Similar to the placement algorithms we proposed in the two vehicular cloud services, one could optimize the placement of the offloading spots to maximize the capacity resulting from the combined vehicle storage. One could adapt the algorithms proposed in the context of the throwboxes work [ZCA+06] to determine the optimal number of offloading spots and their locations to capture the maximum flows of vehicles. We would need to estimate the flows of vehicles between pairs of origin and destination on the road network. A possible way of estimating this origin-destination matrix consists in discretizing the road network space in a collection of areas and using real traffic counts to derive the volume of vehicles traveling between these areas.

Dynamic offloading system. The objective of our work was to assess the feasibility of offloading data over the road network with regard to its capacity and realization. We proposed models and evaluations based on datasets giving for each road segment, real traffic counts averaged over a year. As we discussed in Section 4.5.1, using averaged traffic counts limits the effects of errors due to equipment failures or missing data. As a result, we assumed that vehicles visit the offloading spots following a Poisson distribution, according to the rate we derived from the origin-destination matrix calculation.

In reality, the vehicular traffic is erratic and varies according to seasons, month or time of day. This is expected to affect our offloading system as data has more chances to pile up at offloading spots.To complete our work, one could study the impact of the traffic dynamics with regards to the dimension of the offloading spots and evaluate the resulting capacity. One way to do this is to use the recent work [FS07] that built on Ford and Fulkerson [FJF15] to study the allocation of flows over time using time-expanded graphs.

In the case of data offloading, multiple edges connecting the offloading spots would take into account the discretized variations of the road traffic between the offloading spots (*e.g.*, every 15 minutes). We would then need to estimate the dynamic road traffic between pairs of offloading spots, using work from transportation research [CIM93, BC04]. The dynamic allocation of the data transfers would give a more realistic realization of the offloading infrastructure and would allow us to dimension the system, in particular, the storage needed at the offloading spots to avoid data losses due to overflows.

Comparison with conventional data networks. In our evaluations of the data offloading process, we evaluated its performance in terms of achieved throughput. When evaluating and comparing the offloading process with traditional data networks such as the Internet, several performance metrics such as average throughput, transfer cost, and energy efficiency are relevant to assess the performance of transfers of massive amounts of data. However, the following limitations prevented us from comparing both systems. Firstly, we need to have equivalent settings to plan transfers of bulk data on both sides, including geographical coverage, the amount of data to transfer, the extent of the use of multi-path routing, and current utilization of the network. In particular, we need to be able to use the full capacity of a data network, which varies from one network to the other (*e.g.*, Renater uses mostly 10G links[2] and ESnet uses mostly 100G links[3]), and from one destination to the other (*e.g.*, Paris is better connected than Brest in the Renater network). Secondly, our current implementation of the offloading process does not take into account dynamic road traffic. As a result, the storage needed at the offloading spots cannot be properly provisioned, which hinders the estimate of the cost and energy needed for transferring data. Finally, while some works give estimates of the cost of the bandwidth [LSS+13, JLW+16], the cost of transferring bulk data through an Internet Service Provider (ISP) is not properly defined and varies a lot depending on the ISP and the amount of data to transfer.

6.2.2. Medium- and long-term perspectives

Drayage system. In this thesis, we mainly focused on the feasibility of the offloading system between the edge offloading spots of the data transfers. In particular, we assumed that the transloading of data between the edge offloading spots and endpoints of the data transfers is operated by a drayage system that does not impact the performance of the system.

In Section 3.6, we already discussed ways to realize the drayage system using dedicated lines or dedicated vehicles to transfer the data between the endpoints and the edge offloading spots. Since these solutions are costly and require a dedicated infrastructure, they would not fit with the motivations for this thesis to leverage existing mobility. Instead, we could exploit the work we presented in the extensions to build a drayage system by combining the movements of vehicles that make shorter journeys (*e.g.*, daily commute from or to the workplace).

Combining the mobility of vehicles of different scales would create a multi-tiered architecture involving intermodal end-to-end data transfers, ranging from large-scale (*e.g.*, country-wide) to local (*e.g.*, within a city) data transfers operating seamlessly by the already-existing mobility of the vehicles. Following our extensions, the local data transfers would be done by combining the movements of vehicles carrying the data. The combination could also rely on dedicated facilities (*e.g.*, bus stops) equipped with storage acting as offloading spots to pass the data from one vehicle to another or on specific geographical areas where vehicles come in contact for long durations to transfer large amounts of data.

[2] `https://pasillo.renater.fr/weathermap/weathermap_metropole.html`

[3] `https://www.es.net/`

Offloading spot dematerialization. We dematerialized the offloading spots in the extension of the thesis, such that they represent areas where vehicles can exchange large amounts of data synchronously while in contact. One could implement a similar approach in the context of the vehicular data offloading. Offloading spots could be made floating and the movements of vehicles combined as long as in contact. The data could be then passed directly between vehicles, *i.e.*, without being buffered at stationary intermediate nodes. Note that this approach is similar to the strategies we reviewed in Section 2.2.1.2 that synchronously compose the trajectories of entities at pre-defined locations. The main difference with our approach is that we exploit the capacity available through contact between entities to transfer large amounts of data. To this end, one could leverage the logical representation we used to abstract the mobility of the vehicles (*e.g.*, the offloading overlay). The nodes in the logical representation would refer to the locations where vehicles meet. The logical links would result from the aggregation of the movements of the vehicles and their characterization with networking quantities such as capacity and delay.

The main difference with our approach and those proposed in the state-of-the-art is that a central controller could leverage this logical representation to allocate reliable data transfer. The data would follow a logical path that consists of the vehicles that successively transport the data by exchanging data at the nodes defined in the logical representation. To realize the logical data path, the controller could install states embedded in the vehicles that trigger specific once in contact with other vehicles. These actions are similar to forwarding decisions that can depend on the direction of the encountered vehicles for instance. Finally, the controller could take into account the current and future mobility patterns of the vehicles to dynamically update the logical representation of the system and the forwarding states at the vehicles resulting from the re-allocations of the data transfers.

Vehicle resource virtualization. In the vehicular resource virtualization extension, we virtualized the resources of the vehicles to allow multiple service providers to deploy geo-distributed services that leverage the movements of vehicles. One can combine this work to the logical characterization of the vehicle movements we proposed to manage and allocate by a centralized architecture. We believe that together, these work have the potential to bring original and innovative value-added services for urban environments, especially in the context of the Fog Computing paradigm [BMZA12].

One could involve the use of vehicles equipped with storage, sense, and compute capabilities to provide a densely distributed substrate to help collect and transport sensed data, but also to perform real-time local analysis and decision-making [BMZA12, ASLF14]. To make vehicles operational from the point of view of the Fog Computing paradigm, the mobility of the vehicles has a strong impact on the type of data they generate, aggregate, or collect. One could use our proposal of a controller to allocate and control the resources offered by the vehicles. The controller can help leverage the combined compute and storage capabilities of the vehicles by avoiding the need of collecting data through a control channel (*e.g.*, cellular connection). Instead, the vehicles could aggregate the data they generate or collect from other vehicles they

meet with at dematerialized offloading spots. The vehicles could then process on-the-go the data they carry, while they are moving between the offloading spots. The data, once aggregated and processed by the vehicles, can then be transported to central backend systems for further processing and analysis.

A Traffic modelling techniques

In this chapter, we present the techniques we borrow from transportation research to estimate the origin-destination matrix we used in Chapter 4 to characterize the logical links of the offloading overlay from road traffic counts.

In our evaluations of Chapter 4 and 3 , we used publicly available datasets with Annual Average Daily Traffic (AADT) for the road segments to estimate the road traffic volumes on the paths connecting the offloading spots. Recall that AADT is the total volume of traffic traveling on a road segment in both directions for one year, divided by the number of days in the year. The traffic modelling process follows the steps detailed in the rest of this chapter.

A.1 | Route determination

The first step consists of selecting a subset of the alternative routes connecting each pair of adjacent offloading spots in the road network. Adjacent offloading spots are those within a predefined range of the vehicles as shown in Figure A.1. In the case of electric vehicles, the range is limited to the autonomy of the battery. Current electric vehicles have an autonomy of 300 km on average (*e.g.*, up to 473 km for the Tesla Model S 90D[1], 210 km for the Renault Zoe[2], and 180 km for the Nissan LEAF[3]). In the case of internal combustion engine vehicles, the range is generally higher than electric vehicles. Note that the range of the vehicles highly depends on the driving behavior of the drivers. In the simulations of the following chapters, we considered a conservative vehicle range of 300 km.

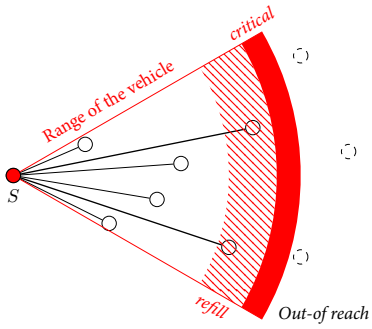


Figure A.1. Illustration of the range of a vehicle and the offloading spots within its range.

The selection of the road paths consists in choosing the paths that are the most relevant to the behavior of the drivers. A natural approach to this problem is to use the All-or-Nothing assignment, which assigns the road traffic to the shortest route between two points [Dij59, DSSW09, HNR68]. While the shortest route may be used by most of the drivers, it does not account for the alternative routes chosen by other drivers, in the case of congestion or road

[1] https://www.Teslamotors.com/models
[2] http://www.renault.fr/gamme-renault/vehicules-electriques/zoe
[3] http://www.nissanusa.com/electric-cars/leaf/charging-range/

work for instance. To determine multiple routes, on can use algorithms that compute the k-shortest routes in terms of travel time [Yen71, Epp98]. However, for large networks, these algorithms yield routes that are most of the time not reasonable and not actually used by the drivers. More recent algorithms try to determine the most reasonable alternative routes between two points [GKS10, ADGW13]. In the evaluations of Chapter 4, we use the algorithm proposed by Abraham *et al.* [ADGW13] to determine the most reasonable alternative routes in the road network of France. With this algorithm, the routes are selected such that they share a low degree of similarity in terms of road segments in common.

A.2 | Route assignment

The second step consists in assigning weights to the selected routes. The weights give the proportion of traffic $p(k)$ that will use a certain route k. If only one route k was selected by a shortest-path algorithm, then, all the road traffic is assigned to this route and $p(k) = 1$. However, if a set of road paths were selected in the route determination step, we determine the proportion of traffic to assign to each selected route using route choice and traffic assignment strategies. These strategies determine weights to the selected routes. The values of the weights are determined according to attributes such as the travel time and the distance of the routes. Those weights reflect the capacity of a route in attracting traffic, the higher the weight of a route, the more traffic it will receive. To this end, stochastic-user-equilibrium (S-U-E) techniques [DS77] were derived from the user-equilibration criterion formulated by Wardrop [War52]. The user-equilibration criterion specifies the following two principles:

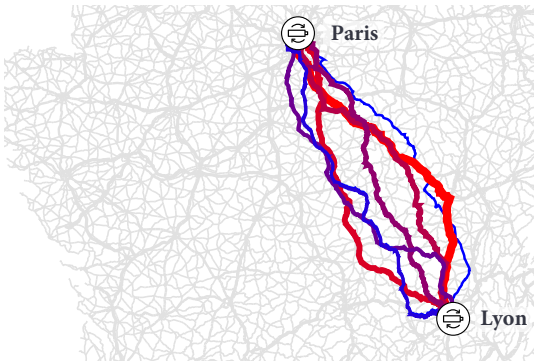


Figure A.2. Top 10 routes between Paris and Lyon chosen with the algorithm proposed by Abraham *et al.* [ADGW13]. The red thickest route is the shortest path and the width of the routes depends on the weights assigned to the routes by the C-Logit traffic assignment [CN96].

1. In an equilibrated network, users cannot improve their travel time by changing routes.
2. The average journey time of all drivers is at a minimum.

The S-U-E techniques build on Wardrop's principles to propose stochastic route assignment algorithms. These techniques include Dial's or C-Logit stochastic traffic assignments [Dia71, CN96]. In the evaluations of Chapter 4, we use the C-logit route assignment model [CN96] to determine the weights on the selected routes in the road network of France. The C-Logit traffic assignment is a multinominal logit model that assigns a choice probability $p(k)$ on a path k using a perceived utility U_k to each path k of the selected paths:

$$U_k = V_k + \varepsilon_k \quad \forall k, \tag{A.1}$$

where V_k is the average or systematic utility of path k (*e.g.*, distance or travel time) and ε_k is the random residual that includes perception errors of the user's decisions. Here, we consider the utility V_k is the sum of the inverse of the travel time of each road segment in the road path k.

The choice probability, or weight, $p(k)$ for path k is expressed as:

$$p\left(k\right) = \frac{\exp\left[V_k - CF_k\right]}{\sum_{h \in \mathcal{P}^{st}} \exp\left[V_h - CF_h\right]}, \tag{A.2}$$

where the term CF_k is the "commonality factor" of path k, which is directly proportional to the degree of similarity of path k with the other selected paths. Here, we consider the following expression of the commonality factor:

$$CF_k = \beta_0 \ln \sum_{h \in \mathcal{P}^{st}} \left(\frac{L_{hk}}{L_k^{1/2} L_h^{1/2}} \right)^{\gamma}, \tag{A.3}$$

where L_{hk} is the length of common paths to h and k, L_h and L_k are the lengths of paths h and k, respectively, and γ is a positive parameter. Typically, $\beta_0 = 1$ and $\gamma = 1$ or $\gamma = 2$ [CN96].

The weights determined by the traffic assignment techniques are then used in combination with the traffic counts to estimate the traffic volume of the routes selected in the first step between each pair of adjacent offloading spots.

A.3 | Trip matrix estimation

Finally, in the third step, we determine the origin-destination trip matrix that characterizes the number of trips between each pair of adjacent offloading spots. This step can be done qualitatively by conducting travel surveys to know the travel habits of the populations within a geographical area. Examples of these national surveys include the National Household Travel Survey (NHTS) in the United States[4] and the Enquête Nationale Transports et Déplacements (ENTD) in France.[5] These surveys are conducted by governmental agencies about every ten years. The last version of the NHTS dates from 2009 while the last version of the ENTD dates from 2008. There is currently a newer version (2016) of the NHTS survey being conducted. A lower bound of an estimate of the road traffic T_{ij} between pairs of locations $(i,\ j)$ corresponds to the traffic volume v_{ab} of the road segment $(a,\ b) \in L^R$ with the lowest traffic of the shortest road path k between the two locations (*i.e.*, that corresponds to the bottleneck of the road path). The volume is weighted by $w(d(k))$, the proportion of trips accounted in travel surveys of the same distance as the one of the shortest road path $d(k)$. This estimate can be expressed as follows:

$$T_{ij} = w(d(k)) \times \min_{(a,\,b) \in k} \{v_{ab}\}, \tag{A.4}$$

[4] `http://nhts.ornl.gov`

[5] `http://www.statistiques.developpement-durable.gouv.fr/sources-methodes/enquete-nomenclature/1543/139/enquete-nationale-transports-deplacements-entd-2008.html` (in French)

Because these surveys take time and are expensive to conduct, there exist other techniques to estimate traffic matrices between selected locations. These techniques usually rely on Annual Average Daily Traffic (AADT). It is important to underline that the AADT is a fundamental statistic used in traffic engineering and transportation planning. The use of the AADT helps reduce the effects of seasonal bias and missing data mainly due to equipment failure, construction schedules, and installation dates that plague continuous traffic monitoring [WHYL97].

In particular, Zuylen and Willumsen [VZW80] proposed an entropy-maximizing formulation to estimate the most likely Origin-Destination (O-D) trip matrix between the defined locations. The objective of the formulation is to maximize the number of ways of selecting an O-D matrix with a total number of trips. As the number of trips increases, the number of ways of selecting an O-D matrix gets more and more peaked and converges to a most likely state. This results in maximizing the entropy of the O-D matrix. The constraint guarantees that the resulting O-D matrix will not overcome the volumes of traffic v_a measured on the road segments a.

Maximize $-\sum_{ij} \left(T_{ij} \log T_{ij} - T_{ij}\right)$,

subject to

$v_a - \sum_{ij} T_{ij} p_{ij}^a = 0 \qquad \forall a \in L^R.$

The probability p_{ij}^a denotes the route choice probability of route between source i and destination j to take link a among S_{ij}, the set of selected road path. This probability is derived from the choice probability and computed in the route assignment as follows:

$$p_{ij}^a = \sum_{\substack{k \in S_{ij} \\ k \ni a}} p(k), \tag{A.5}$$

Step 1. Initialization
- $\lambda_a \leftarrow 0 \quad \forall a \in L$

Step 2. Iterative balancing
- **repeat**
  - $T_{ij}^\star \leftarrow \exp\left(\sum_a \lambda_a \, p_{ij}^a\right) \quad \forall i, \, j$
  - **forall** constraints (over domain L^R) **do**
    - **if** $V_a \neq \sum_{ij} T_{ij}^\star \, p_{ij}^a$ **then**
      - $\lambda_a \leftarrow \lambda_a + \ln(V_a) - \ln\left(\sum_{ij} T_{ij}^\star \, p_{ij}^a\right)$
- **until** Convergence

Algorithm A.1: Iterative balancing algorithm for the entropy maximization problem.

Since the optimization problem presented above is convex, the authors presented an iterative balancing algorithm to solve it, derive from Bergman's method to solve such problems [Bre67].

The algorithm is presented in Algorithm A.1. Each balancing iteration adjusts the quantity λ_a, the road traffic assigned to each road segment a. This adjustment is done until all the constraints on the volumes of the road segments are satisfied.

With the traffic modelling techniques presented in this appendix, we were able to derive the traffic volumes on the road paths connecting the offloading spots from traffic counts on road segments. In the vehicle flow allocation problem of Chapter 4, we used these traffic volumes to characterize the logical links of the offloading overlay that connect offloading spots together.

B Résumé de la thèse en français

B.1 | Introduction

B.1.1. Contexte

La popularité croissante des applications demandeuses de données pousse les fournisseurs de contenus à trouver de nouvelles approches pour répliquer et synchroniser de larges quantités de données entre des répertoires distants. Ainsi, les fournisseurs de contenus s'intéressent de plus en plus aux techniques de délestage de données (*offloading* en anglais) puisqu'elles représentent une solution peu coûteuse pour *étendre* la capacité d'un réseau de données. Le délestage de données vise à exploiter des médiums de transmission alternatifs et de nouveaux modèles de livraison de données [Pat03].

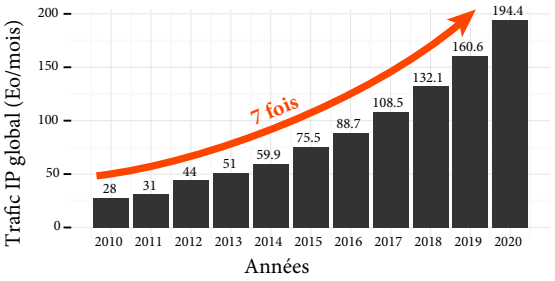


Figure B.1. Trafic IP global [Cis16].

De nombreux fournisseurs de contenus utilisent déjà des techniques de délestage de données pour livrer du contenu. C'est le cas de Netflix, qui a commencé en 1998 à offrir un service de location de films à ses clients en leur envoyant des DVDs par voie postale.[1] Il y a quelques années, Amazon a développé le service Import/Export qui permet à ses clients d'envoyer leurs données dans des disques durs par voie postale directement à Amazon, alors en charge de réceptionner les données et de les mettre à disposition dans ses bases de données S3.[2] Ce service est particulièrement utile pour les clients de Amazon limités par leur connexion Internet à faible débit. Plus récemment, afin de systématiser ces transferts postaux, Amazon a présenté le Amazon Snowball, un appareil portatif prêt à l'envoi et disposant de 50 TB de stockage pour transporter plusieurs Pétaoctets de données vers les bases de données de Amazon. L'ensemble de ces techniques de délestage ont pour but de s'affranchir des limitations actuelles pour transférer de grandes quantités de données, telles que les coûts, la durée ou la sécurité de tels transferts.

Dans cette thèse, nous proposons d'équiper des véhicules particuliers de capacités de stockage de données pour transformer le réseau routier en un système de transmission à large capacité. Seulement 10% des véhicules en circulation en France équipés de disques durs de 1 Téraoctets

[1] `http://dvd.netflix.com/`

[2] `https://aws.amazon.com/fr/importexport/`

ont le potentiel de transporter 115 Exaoctets de données par jour (ce qui équivaut à 1,3 Petaoctets par seconde). Élargi aux 1,2 milliards de véhicules en circulation dans le monde, la mobilité quotidienne des véhicules représente un potentiel inexploité pour faire face à l'avalanche de données sur les réseaux de données traditionnels tels que l'Internet qui approche. Comme montré sur la Figure B.1, les échanges de données ont explosé en triplant ces six dernières années et est prévu de doubler ces cinq prochaines années [Cis16, GR12, Hec16].

L'utilisation opportuniste des véhicules particuliers suit la tendance de services qui reposent sur l'économie partagée tels que Uber ou BlaBlaCar en mettant à disposition le partage de biens et de services de pair à pair. Dans notre cas, les trajets effectués par les véhicules sont les ressources partagées que l'on met à disposition des fournisseurs de contenus pour transporter leurs données.

Équiper les véhicules de capacités de stockage de données suit également une tendance actuelle qui crée des services à valeur ajoutée à partir d'un service "de base". C'est le cas de services de livraison de colis proposés en complément de services de transportation de passagers. Par exemple, l'américain Greyhound propose un tel service qui repose sur les trajets existants des cars effectués dans le cadre du transport de passagers.[3] De la même manière, l'américain Uber propose UberRUSH, un service de livraison de colis à la demande qui se repose sur une flotte de véhicules normalement dédiés au transport de personnes.[4]

> *Dans cette thèse, nous caractérisons et exploitons la mobilité existante d'entités de tous les jours pour atténuer ou s'affranchir des limitations des réseaux de données traditionnels tels que l'Internet. Nous repensons ainsi la conception de services populaires tels que les transferts de données en masse ou les services de partage de données de type "cloud" en limitant ou en s'affranchissant de leur dépendance vis-à-vis de ces réseaux.*

B.1.2. Vision et conception du service de délestage de données

Nous exploitons la tolérance aux délais du trafic d'arrière-plan pour délester de grandes quantités de données des réseaux de données conventionnels tels que l'Internet vers le réseau routier. En particulier, nous nous intéressons aux transferts de données qui ont lieu dans le cadre d'applications avec une tolérance aux délais de plusieurs jours (par exemple, la distribution de contenus scientifiques ou le trafic d'arrière-plan résultant des activités de maintenance et d'approvisionnement de systèmes distribués large-échelle [LSRS09]). Notre délestage de données exploite de manière opportuniste le nombre croissant de trajets de véhicules pour physiquement transporter des données [LJ10].

Une façon simple de réaliser le délestage de données consiste à utiliser les véhicules qui font l'intégralité du trajet entre la source et la destination du transfert de données. Cependant, comme montré sur la Figure B.2, moins de 1,4% des trajets effectués par des véhicules en France

[3] `http://www.shipgreyhound.com/`

[4] `https://rush.uber.com`

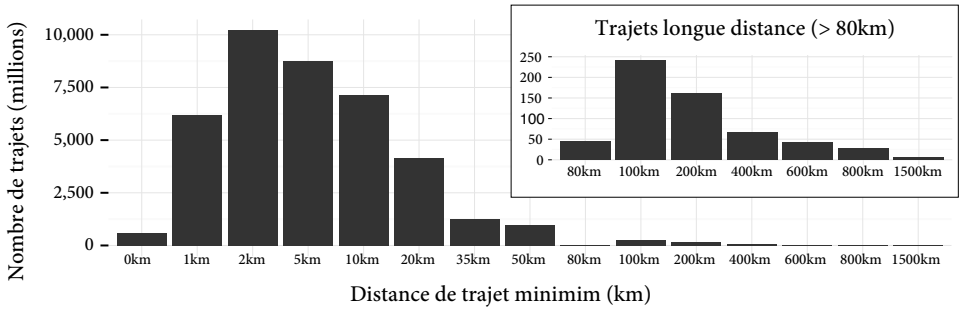


Figure B.2. Distribution des trajets par distance parcourue par les véhicules en France. Source : Enquête Nationale Transports et Déplacements (ENTD), 2008 [MED].

sont long d'au moins 80 km. Bien que cette valeur soit tout de même conséquente pour des transferts sur de courtes distances (cela correspond à environ 600 millions de trajets par an), ce nombre décroit fortement en augmentant la distance des transferts. De plus, en augmentant la portée des transferts, le nombre de destinations différentes augmente, réduisant le nombre de trajets et en laissant juste quelques uns pour livrer les données sur de longues distances. Une alternative consiste à utiliser une flotte de véhicules dédiés au transport de données de la source du transfert à sa destination. Bien que cette solution n'emploie pas la mobilité existante des véhicules, elle est coûteuse à mettre en place et ne passe pas à l'échelle lorsque le nombre de transferts et de destinations augmente, puisque de plus en plus de véhicules dédiés sont nécessaires.

À la place de ces deux solutions, nous exploitons des facilités dédiées pour composer plusieurs flots de véhicules effectuant des trajets vers différentes directions. Tout au long de cette thèse, nous appellerons ces facilités des *points de délestage*, comme représentés sur la Figure B.3. Ces points de délestage sont équipés de capacités de stockage pour stocker de manière temporaire les données provenant de véhicules s'y arrêtant et ainsi relayer ces données entre véhicules appartenant à différents flots. Ils sont placés aux endroits où les véhicules s'arrêtent fréquemment et suffisamment longtemps pour transférer de grandes quantités de données. Ces endroits correspondent aux emplacements de parking situés le long des routes ou ceux situés dans des villes ou des supermarchés, ou encore des stations-services ou de chargement de batteries électriques.

À partir de notre travail initial concernant le délestage de données au cœur de cette thèse, nous étendons le concept de point de délestage selon deux directions dans le cadre de services véhiculaires de type "cloud". Dans une première extension, nous exploitons la capacité de stockage des points de délestage pour les transformer en dépôts de fichiers et mettre à disposition un système géo-distribué de stockage et de partage de fichiers pour utilisateurs mobiles. Afin d'assurer le partage de fichiers, nous exploitons les mouvements des véhicules entre les points de délestage pour répliquer les fichiers et les rendre accessibles aux utilisateurs. Dans une seconde extension, nous virtualisons les ressources des véhicules afin de créer un réseau véhiculaire mobile. Nous dématérialisons les points de délestage en zones prédéfinies caractérisées par une densité suffisamment élevée de véhicules pour transférer des machines virtuelles entre véhicules. Par

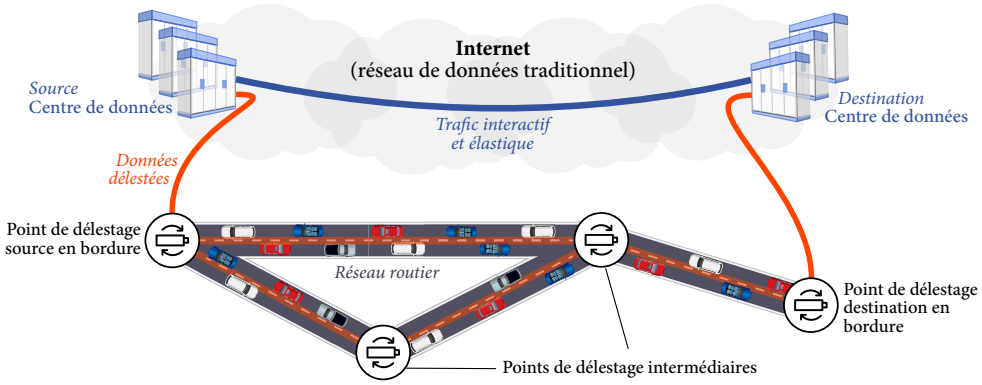


Figure B.3. Vue générale du délestage de trafic que nous proposons dans cette thèse.

ces extensions, nous montrons que la mobilité existante des véhicules a le potentiel de réduire les limitations et la dépendance vis-à-vis des réseaux de données traditionnels.

B.1.3. Description des problèmes rencontrés

Dans un premier temps, il nous faut allouer de manière efficiente des demandes de délestage de transferts de données aux flots de véhicules existants. Dans la suite de la thèse, nous appelons ce problème le *problème d'allocation de flots de véhicules*. Afin de le résoudre, nous devons créer un processus efficient d'allocation des transferts de données sur des flots de véhicules tel qu'il optimise l'utilisation de ces flots (ressources), tout en satisfaisant les besoins des demandes de délestage correspondantes.

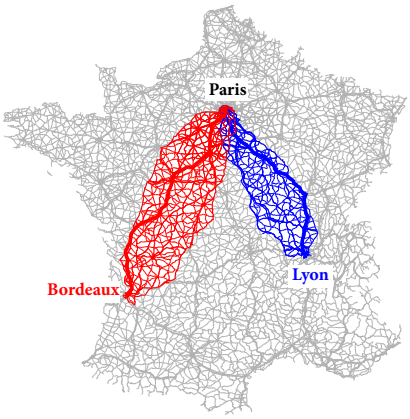


Figure B.4. Routes principales reliant Paris à Lyon et Bordeaux.

Dans un deuxième temps, il nous faut maîtriser l'échelle du réseau routier afin de pouvoir effectuer un délestage de données qui passe à l'échelle de manière efficiente. En effet, la complexité de la topologie du réseau routier et le grand nombre de trajets effectués rendent le problème d'allocation de flots de véhicules insoluble par calcul. Nous illustrons le besoin d'un mécanisme d'allocation qui passe à l'échelle avec la Figure B.4, où nous représentons les routes des trajets potentiels qui peuvent êtres alloués pour délester des transferts de Paris vers Bordeaux et Lyon.

Finalement, nous devons assurer des transferts de données fiables. Parce que les véhicules peuvent ne pas délivrer les données qu'ils transportent vers le prochain point de délestage ou la destination finale, nous devons utiliser et adapter des techniques pour recouvrer des données perdues.

Dans les extensions du travail initial, le problème commun est de déterminer les emplacements des points de délestage, selon qu'ils soient matérialisés ou pas, et selon les besoins spécifiques des services que nous déployons.

Plus particulièrement, dans le système de partage de fichiers, nous devons concevoir un algorithme pour placer les points de délestage jouant le rôle de dépôts à des emplacements stratégiques de telle façon qu'ils capturent un nombre maximal de requêtes utilisateurs avant qu'elles n'expirent, tout en les connectant avec les mouvements des utilisateurs pour mettre en place la réplication des fichiers au niveau des dépôts. Dans le cas du réseau véhiculaire virtuel, nous devons identifier les emplacements où les véhicules se rencontrent fréquemment et suffisamment longtemps pour transférer de grandes quantités de données, telles que des machines virtuelles.

B.2 | *Délestage massif de données sur le réseau routier*

Dans ce chapitre, nous proposons d'évaluer l'utilisation des véhicules circulant sur le réseau routier comme technique de délestage, comme illustré sur la Figure B.5. Les véhicules en circulation sont assimilés aux ressources disponibles dans le réseau routier. Les demandes de délestage provoquent des transferts de données, chacun caractérisé par le délai d'acheminement attendu et la quantité de données à transférer. Nous formulons le problème d'allocation de flots de véhicules en allouant les ressources du réseau routier selon une politique tenant compte des contraintes exprimées en termes de délai et de débit qui caractérisent ce transfert. La définition d'une politique d'allocation des ressources est un problème NP-complet et les techniques traditionnelles de réduction ne sont pas applicables en raison de la complexité de la topologie routière et du nombre de véhicules circulant sur les routes.

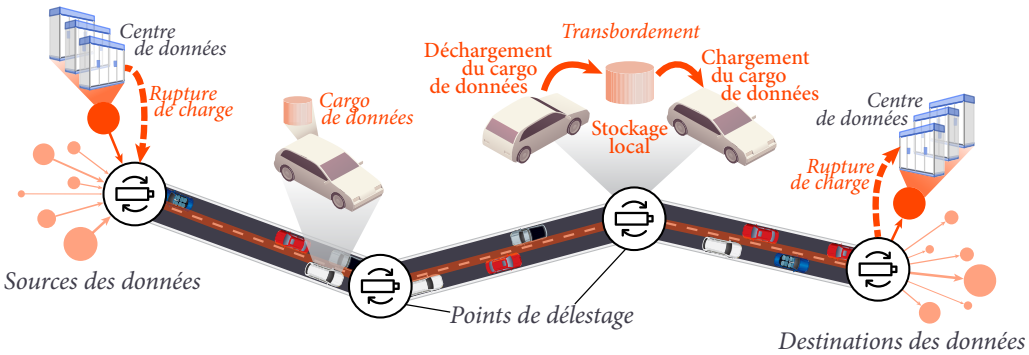


Figure B.5. Overview of the operations to offload large amounts of data over the road network.

Afin de contourner cette complexité, nous employons une approche en deux phases. D'une part, nous simplifions le réseau routier en construisant un réseau de recouvrement, appelé "réseau de délestage" qui offre une vue logique du réseau routier sous-jacent (Figure B.8). En éliminant un grand nombre de routes inutiles, la complexité due à la taille du réseau est mieux maîtrisée. D'autre part, nous proposons une formulation de l'allocation des flots de véhicules par un algorithme max-min équitable utilisant des modèles de programmation linéaire pour déterminer les chemins logiques optimaux pour le transport des données délestées (Section B.2.4). Enfin, nous évaluons la capacité de notre système de délestage sur l'infrastructure routière française avec des données de trafic routier réelles (Section B.2.5).

B.2.1. Architecture centralisée pour le délestage de données utilisant les véhicules

Dans cette section, nous donnons une vue globale des opérations de l'infrastructure en charge de délester les données. Nous nous intéressons à des transferts qui durent plusieurs jours engendrés par des activités de maintenance ou d'approvisionnement pour des migrations de machines virtuelles ou des sauvegardes hors-ligne entre centres de données (*data centers* en anglais). La figure B.5 montre les opérations de notre infrastructure de délestage sur un fragment de réseau routier.

Des véhicules particuliers sont équipés de disques durs leur permettant de stocker des données. De plus, les véhicules sont équipés d'interfaces de communication pour échanger des données à haut débit.

Le flot de véhicules ainsi équipés joue le rôle d'un *backhaul* mécanique qui connecte une collection de points de délestage, comme montré sur la Figure B.5. Les points de délestage ont deux rôles selon leur position relative dans le processus de délestage. Ce double rôle est montré sur la Figure B.5. Une partie ou tout le trafic de données provenant d'un réseau de données traditionnel est premièrement *dévié* de ce réseau vers le plus proche point de délestage en bordure du transfert (rupture de charge). Les données sont temporairement stockées dans le point de délestage et chargées sur les véhicules s'y arrêtant. Les points de délestage suivants sont alors en charge de relayer les données en les *transbordant*. Les véhicules déchargent les données qu'ils transportent pendant leur arrêt aux points de délestage. Les données sont temporairement stockées jusqu'à ce qu'un véhicule ne les charge lors de son arrêt et les transporte jusqu'au point de délestage suivant. Une fois arrivées au point de délestage destination en bordure, les données sont à nouveau déviées vers le réseau de données traditionnel.

Pour implémenter le délestage de données sur le réseau routier, nous exploitons les avantages d'une centralisation logique en utilisant une architecture de type SDN (*Software-Defined Networking*) qui permet un contrôle efficient et une gestion de l'infrastructure de délestage. En effet, le concept de SDN fournit les moyens logistiques, en particulier la planification, l'orchestration, l'implémentation et lecontrôle pour mettre en place le délestage de données [MAB+08].

L'architecture centralisée comprend un contrôleur central et une collection de points de délestage qui agissent comme des commutateurs en acheminant les données.

Afin d'assurer le fonctionnement en continu du service de délestage, le contrôleur surveille le fonctionnement des points de délestage, en particulier la quantité d'espace de stockage disponible et les destinations des données en transbordement. Pour améliorer les décisions d'acheminement des données en prédisant la future destination des véhicules, le fournisseur de service récupère l'itinéraire suivi par les véhicules qui s'arrêtent aux points de délestage via leur système de navigation.

Le contrôleur reçoit les demandes pour délester des données et est en charge d'assigner les transferts à des chemins routiers et les flots de véhicules correspondants. Un chemin routier correspond aux flots de véhicules successifs qui connectent une séquence de points de délestage. Le contrôleur détermine ces chemins en résolvant le problème d'allocation de flots de véhicules

qu'on formule avec un modèle max-min équitable dans la Section B.2.4. Cette formulation maximise le débit résultant de l'allocation des transferts de données tout en la garantissant équitable afin d'éviter les famines pour les transferts alloués. Le résultat du problème est ensuite utilisé pour configurer la séquence de points de délestage participant aux transferts alloués.

B.2.2. Modèle économique avec les véhicules électriques

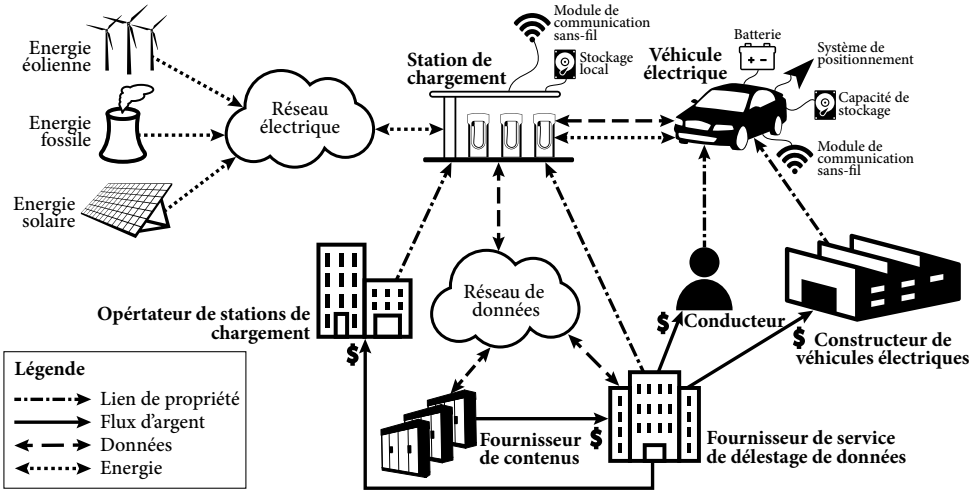


Figure B.6. Modèle économique du délestage de données.

Dans cette section, nous présentons le le modèle économique que l'on utilisera dans la suite de cette thèse. Ce modèle inclut (*i*) des véhicules électriques et leur conducteurs, (*ii*) un fournisseur de service de délestage de données (qui peut être le constructeur de véhicules électriques), (*iii*) un opérateur de stations de chargement de batteries de voitures électrique, et enfin (*iv*) un fournisseur de contenus qui opère plusieurs centres de données géo-distribués.

En particulier, ce modèle nous permet de modéliser et analyser ce que nous proposons dans des conditions réalistes en se basant sur des éléments qui nous permettent de réaliser le délestage de données sur le réseau routier tels que la portée des véhicules électriques qui permet d'avoir des arrêts fréquents ou le temps de chargement de batteries électriques qui permet de transférer de grandes quantités de données pendant le temps de chargement de la batterie.

Dans le cadre de ce scénario, les véhicules électriques sont équipés de capacités de stockage de données et les stations de chargement de batteries électriques jouent le rôle de points de délestage permettant de charger ou décharger les données des véhicules pendant la charge de leur batterie. Le fournisseur du service de délestage utilise et gère le contrôleur central présenté dans la section précédente.

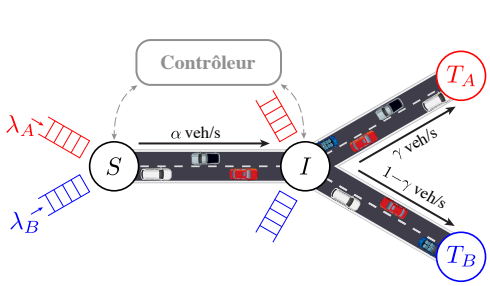
Le fournisseur de service de délestage de données, si différent du constructeur de véhicules électriques, propose des compensations financières aux conducteurs pour effectuer leurs trajets habituels à condition de transporter un disque dur à bord de leur véhicule. La compensation financière est déterminée avec l'opérateur de stations de chargement de batteries électriques

en fonction du nombre de kilomètres parcourus et de la couverture effective. Dans le cas où le constructeur de véhicules électriques a le rôle du fournisseur de service de délestage de données, les véhicules sont équipés de série avec des disques durs et le service est fourni sans compenser les conducteurs. Le fournisseur de service de délestage de données facture le fournisseur de contenus en fonction de la quantité de données délestée sur le réseau routier et partage les revenus avec l'opérateur de stations de chargement.

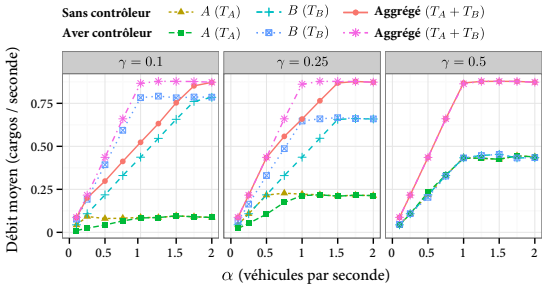
Une fois les demandes de délestage de transferts de données émises par des fournisseurs de contenus, le fournisseur de service de délestage détermine la quantité de données à allouer à chaque flot de véhicules se déplaçant entre les points de délestage.

B.2.3. Motivation pour avoir un contrôleur centralisé

Dans cette section, nous présentons un exemple de réseau routier volontairement simple pour montrer les bénéfices d'un contrôleur centralisé. Le réseau routier est représenté sur la Figure B.7a et comporte quatre points de délestage. Nous comparons deux stratégies d'ordonnancement implémentées au niveau du point de délestage S. Nous considérons deux transferts de données A et B provenant tous deux de S, partageant tous deux le tronçon I et allant à leurs destinations finales respectives T_A et T_B. Nous considérons seulement le trafic routier des tronçons représentés sur la figure.



(a) Deux transferts de données A et B provenant tous deux de S partagent le tronçon I et suivent leur route jusqu'à leurs destinations respectives T_A et T_B.



(b) Débit maximal (exprimé en nombre de cargos de données délivrés par seconde) pour les deux stratégies d'ordonnancement en fonction des deux paramètres (α, γ).

Figure B.7. Réseau routier composé de quatre points de délestage afin de montrer les bénéfices résultant d'un contrôleur.

Nous considérons deux stratégies d'ordonnancement implémentées au niveau du point de délestage S: (*i*) sans contrôleur et (*ii*) avec contrôleur. Elles permettent de sélectionner les cargos de données issus d'un transfert de données (soit A, soit B) à charger sur chaque véhicule s'arrêtant au point de délestage S. La première stratégie (sans contrôleur) suit un ordonnancement de type "Round-Robin" en sélectionnant les données dans les mêmes proportions et en ordre circulaire, sans donner de priorités aux transferts. La seconde stratégie (avec contrôleur) utilise un contrôleur pour définir des configurations à installer au niveau du point de délestage S pour effectuer la décision pondérée de sélection du transfert et du cargo de données à charger dans

le véhicule à l'arrêt. Le contrôleur détermine cette configuration avec des poids proportionnels aux volumes de trafic routier sur les différents tronçons. Ainsi, si l'on considère un trafic routier trois fois plus important sur le tronçon $(I,\, T_A)$ par rapport au tronçon $(I,\, T_B)$, la seconde stratégie va charger trois fois plus de données sur les véhicules en direction de T_A. Ce n'est pas le cas de la première stratégie : celle-ci va charger les mêmes quantités de données sur les véhicules qu'ils aillent en direction de T_A ou de T_B.

Nous utilisons SUMO pour évaluer les deux stratégies d'ordonnancement [BBEK11]. À l'aide de ce simulateur, nous simulons le trafic routier avec des volumes de (*i*) α véhicules par seconde de S vers I, (*ii*) γ de I vers T_A, et (*iii*) $1-\gamma$ de I vers T_B. Nous supposons que chacun des deux transferts A et B a une quantité de données infinie à transporter de S vers T_A et T_B, respectivement (*i.e.*, $\lambda_A = \lambda_B = \infty$). La Figure B.7b montre le débit du système mesuré en terme de nombre de cargos de données livrés à la destination par seconde.

Nous constatons qu'avec la première stratégie (sans contrôleur), le transfert A a besoin de plus de débit routier (α) pour atteindre son débit nominal. Sans connaissance du trafic routier en aval, S ne peut pas charger les quantités optimales de données au prochain point de délestage de telle façon que I utilise pleinement les ressources véhiculaires. De ce fait, I n'a pas assez de données disponibles localement pour alimenter le flot routier à destination de T_B. Les ressources disponibles sur le tronçon $(I,\, T_B)$ restent sous-utilisées si les cargos de données ne sont pas transportés en nombre adéquat de S vers I (par exemple, $\alpha = 2$). Au contraire, la seconde stratégie avec contrôleur permet de transporter un nombre adéquat de cargos de données de S vers I. De ce fait, I a suffisamment de données à disposition pour utiliser pleinement les véhicules en direction de T_A et T_B. Le contrôleur utilise sa connaissance du trafic routier en aval pour maximiser l'utilisation des ressources au niveau du point de délestage S. Cette stratégie donne un meilleur débit pour chacun des transferts en compétition au niveau de S.

À l'aide de cet exemple simple, nous avons montré les bénéfices d'une architecture centralisée qui pré-configure les points de délestage afin d'effectuer des décisions pondérées pour charger les données sur les véhicules à l'arrêt. Cette architecture permet d'avoir de meilleurs débits par rapport à une architecture distribuée où les décisions de chargement de données sur les véhicules sont faites de manière locale.

B.2.4. Allocation des flots de véhicules pour des transferts de données délestés sur le réseau routier

Le problème d'allocation de flots de véhicules revient à déterminer les routes suivies par les données transportées par les flots de véhicules entre les points de délestage. Avant de résoudre ce problème, il nous faut nous affranchir de la complexité du réseau routier de par la taille et la couverture de sa topologie et de par le nombre de trajets véhiculaires qui rendent le problème insoluble par calcul.

Dans cette section, nous présentons un algorithme pour réduire la complexité du réseau routier. Le résultat de cet algorithme est un réseau de délestage en sur-couche qui est une représentation

logique du réseau routier et en capture ses principales caractéristiques qui nous sont nécessaires pour résoudre le problème d'allocation de flots de véhicules.

B.2.4.1 Notations préliminaires

Réseau routier. Nous représentons le réseau routier par un graphe orienté composé de tronçons de routes (les liens) et d'intersections (les nœuds). Les points de délestage correspondent également à des nœuds et sont situés aux intersections ou le long des tronçons. Pour un tronçon $(a,\, b)$, nous définissions par v_{ab} le volume nominal de véhicules (en terme de nombre de véhicules par unité de temps), par c_{ab} la capacité du tronçon (également en terme de nombre de véhicules par unité de temps), et par $t_{ab}(v_{ab})$ le temps de trajet, fonction du volume nominal de véhicules.

Nous utilisons des jeux de données qui comportent les volumes de trafic sur les tronçons de route exprimés par des Trafic Moyens Journaliers Annuels (TMJA). Le TMJA est le volume de trafic total mesuré traversant un tronçon de route dans les deux sens pendant une année divisé par le nombre de jours dans l'année. De ce fait, la capacité c_{ab} du tronçon $(a,\, b)$ est fixée à la valeur du TMJA pour ce tronçon dans le jeu de données.

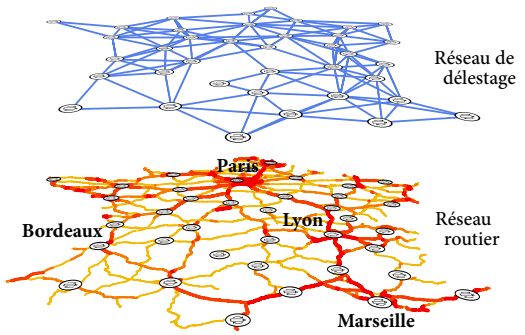


Figure B.8. Le réseau de délestage représenté au dessus du réseau routier français. Le réseau de délestage résulte d'un déploiement de stations de chargement de batteries électriques couvrant le réseau routier français.

Réseau de délestage en surcouche. Afin de nous affranchir de la complexité du réseau routier, nous proposons un algorithme de réduction du réseau routier. Notre algorithme construit un réseau de recouvrement, appelé *réseau de délestage*, qui offre une vue logique simplifiée du réseau routier sous-jacent. Le réseau de recouvrement est représenté par un graphe orienté noté G^O, où les nœuds N^O sont les points de délestage. Les liens logiques L^O qui connectent les points de délestage résultent de l'agrégation des tronçons du réseau routier G^R sous-jacent. Un lien logique $(i,\, j)$ de L^O est caractérisé par sa capacité $c(i,\, j)$ calculée en fonction du trafic routier, son délai de propagation $t(i,\, j)$ qui représente le temps de trajet moyen le long des routes sous-jacentes et par un taux d'erreur $l(i,\, j)$. Le taux d'erreur correspond à la proportion des véhicules n'arrivant pas à destination en raison de prédictions erronées de la future direction des véhicules lors de leurs arrêts aux points de délestage.

B.2.4.2 Caractérisation du réseau de délestage en surcouche

Dans cette section, nous caractérisons les liens logiques du réseau de délestage en surcouche par des quantités réseau afin de résoudre le problème d'allocation de flots de véhicules.

Temps de trajet $t(i,\, j)$**.** Le temps de trajet t_{ab} du tronçon $(a,\, b)$ est donné par la fonction définie par le BPR (*Bureau of Public Roads* aux États-Unis) par :

$$t_{ab}(v_{ab}) = t_{ab}(0)\left[1 + \alpha\left(\frac{v_{ab}}{c_{ab}}\right)^{\beta}\right], \tag{B.1}$$

où α et β sont des paramètres définis par le BPR qui dépendent du profil de la route (typiquement $\alpha = 0.15$ minutes et $\beta = 4.0$) [Nat00].

De l'Équation B.1, nous pouvons en déduire le temps de trajet sur le chemin routier p, noté t_p, qui correspond à la somme des temps de trajet des tronçons qui composent le chemin (sans considérer les délais aux intersections) :

$$t_p = \sum_{(a,\, b)\in p} t_{ab}(v_{ab}). \tag{B.2}$$

De l'Équation B.2, nous en déduisons une expression du temps de trajet $t(i,\, j)$ sur les r chemins routiers entre i et j, pondéré par les volumes de trafic routier v_p sur chacun des chemins p.

$$t(i,\, j) = \frac{\sum_{p\in S_{ij}} t_p\, v_p}{r \sum_{p\in S_{ij}} v_p}, \tag{B.3}$$

où S_{ij} correspond à l'ensemble des chemins routiers entre i et j, sans cycles.

Capacité $c(i,\, j)$**.** La capacité $c(i,\, j)$ du lien logique $(i,\, j) \in L^O$ est fonction de la somme des volumes de trafic v_p des chemins routiers entre les points de délestage i et j (*i.e.*, le nombre de véhicules par unité de temps allant de i à j sur le chemin routier p). La capacité du lien logique est également fonction du taux de pénétration de marché $\mathcal{M}$ des véhicules participant au délestage de données et de la capacité de stockage des véhicules $\mathcal{S}$ (nous supposons que tous les véhicules sont équipés de disques durs de même capacités $\mathcal{S}$).

$$c(i,\, j) = \mathcal{M}\mathcal{S} \sum_{p\in\mathcal{P}^{ij}} v_p. \tag{B.4}$$

Dans l'évaluation des performances présentée dans la Section B.2.5, nous déduisons la valeur de v_p pour chacun des chemins p à partir du jeu de données du réseau routier français. Ce jeu de données contient des comptages de trafic routier réels pour chacun des tronçons dont chaque chemin p est constitué.

Taux d'erreur $l(i,\, j)$**.** Le taux d'erreur $l(i,\, j)$ (compris entre 0 et 1) du lien logique $(i,\, j)$ représente la proportion de données perdues entre les points de délestage i et j. Comme mentionné précédemment, ces pertes sont dues aux erreurs de prédictions des futures directions des véhicules lors de leurs arrêts aux points de délestage.

Demandes de délestage. Une demande de délestage d_{st} entre deux points de délestage s et t est caractérisée par la quantité de données à transférer β_{st} et un délai imparti pour transporter ces données τ_{st}. Par simplification, nous modélisons le taux des demandes arrivant à la source s par une distribution de Poisson λ_{st} avec un débit moyen de β_{st}/τ_{st}.

Pour améliorer la lisibilité, nous donnons les notations utilisées dans la Table B.1.

Table B.1: Table des notation utilisées pour l'algorithme d'allocation max-min équitable.

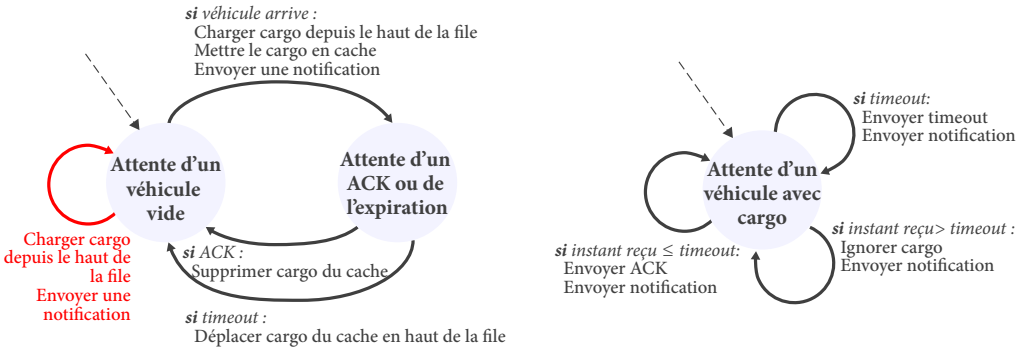
Variable	Signification
d_{st}	Demande de délestage entre la source s et la destination t
τ_{st}	Temps imparti pour transférer les données de la demande de délestage d_{st}
β_{st}	Quantité de données à transférer pour la demande de délestage d_{st}
λ_{st}	Taux d'arrivées Poissonniennes à la source s de la demande de délestage d_{st}
$\mathcal{P}_{st}$	Ensemble de chemins logiques entre s et t sur le réseau de délestage
$\mathcal{S}$	Capacité de stockage des véhicules
$\mathcal{M}$	Taux de pénétration de marché
$c\,(i,\,j)$	Capacité du lien logique $(i,\,j)$
$t\,(i,\,j)$	Temps de trajet sur le lien logique $(i,\,j)$
$l\,(i,\,j)$	Taux d'erreur du lien logique $(i,\,j)$
$l_{st}^{\textbf{red}}\,(i,\,j)$	Taux d'erreur sur le lien logique $(i,\,j)$ avec redondance pour la demande de délestage d_{st}
$O_{st}^{\textbf{red}}$	Quantité additionnelles transmises dues au mécanisme de **red**ondance sur le flot de données pour la demande de délestage d_{st}
$R_{st}^{\{\text{hbh,e2e}\}}(i,\,j)$	Nombre moyen de transmissions (soit de bout-en-bout (**e2e**) ou saut-à-saut (**hbh**) pour un cargo de données sur le lien logique $(i,\,j)$ pour la demande de délestage d_{st}
δ_i	Temps d'attente au point de délestage i (pour recharger la batterie du véhicule électrique)
$f\,(p)$	Flot de données sur le chemin logique p
$t\,(p)$	Temps de trajet du chemin logique p
$O_{st}(i,\,j)$	Données additionnelles pour la demande de délestage d_{st} sur le lien logique $(i,\,j)$
$R_{st}(i,\,j)$	Nombre de transmission d'un cargo de données sur le lien logique $(i,\,j)$ indépendamment du mécanisme de retransmission utilisé

B.2.4.3 Fiabilité des transferts délestés

Dans cette section, nous présentons les deux mécanismes que nous utilisons pour fiabiliser les communications et leur impact sur la quantité de données additionnelles à transférer (*overhead* en anglais).

Les taux d'erreur des liens empruntés sont absorbés pour obtenir des transferts fiables en appliquant les deux méthodes suivantes : (*i*) encodage des données selon des techniques de suréchantillonnage qui procèdent par ajout de redondance (par exemple, RAID) et (*ii*) retransmission des données perdues sur un lien logique depuis le point de délestage précèdent (par exemple, SR-ARQ [LC04]). Ces deux méthodes augmentent la quantité de données à transporter par un facteur multiplicatif noté respectivement o_{st}^{red} pour les techniques de redondance et o_{st}^{ret} pour celles basées sur la retransmission des données.

Redondance des cargos de données. L'utilisation de la redondance permet néanmoins de réduire les pertes sur chaque lien logique, dont le taux de perte passe de $l(i,\,j)$ à $l_{st}^{\text{red}}(i,\,j)$. Nous



(a) Côté émetteur (point de délestage i). **(b)** Côté récepteur (point de délestage j).

Figure B.9. Retransmissions avec SR-ARQ. Les transitions en rouge sont exclusivement pour les points de délestage intermédiaires dans le cas de la retransmission de bout-en-bout (**e2e**). Les autres transitions sont pour le point de délestage source dans le cas de retransmissions de bout-en-bout (**e2e**), et ceux source et intermédiaires dans le cas de retransmission de saut-à-saut (**hbh**).

utilisons la redondance avec RAID 6 qui arrange les cargos de données en ensembles de n cargos ($n \geq$), incluant deux cargos redondants (et $n-2$ cargos utiles). Le nouveau taux de perte $l_{st}^{\text{red}}(i,\, j)$ du lien logique $(i,\, j)$ est fonction du nombre n de disques durs transportés par les véhicules traversant le lien logique $(i,\, j)$ et arrangés ensembles avec RAID 6 :

$$l_{st}^{\text{red}}(i,\, j) = 1 - \sum_{k=0}^{2} \binom{n}{k} l(i,\, j)^k \big(1 - l(i,\, j)\big)^{n-k}. \tag{B.5}$$

Cette expression suppose un taux de perte équivalent à la probabilité de panne d'un disque dur dans le cas d'utilisation de RAID avec des systèmes informatiques, ce qui est consistant puisque les deux cas sont indépendants et identiquement distribués.

La redondance avec RAID 6 ajoute des données additionnelles à transmettre d'un facteur o_{st}^{red}. Pour un transfert de N arrangements RAID 6 de chacun n cargos de données, le nombre de données additionnelles à transmettre est exprimé par :

$$o_{st}^{\text{red}} = \frac{n}{n-2}. \tag{B.6}$$

Retransmission des cargos de données. Nous proposons deux types de retransmission de type SR-ARQ (*Selective Repeat Automatic ReQuest* ou requête automatique avec répétition sélective) des cargos de données, en plus de la redondance des cargos arrangés avec RAID 6, puisque cette redondance ne peut pas récupérer toutes les pertes de données. D'une part, nous considérons des retransmissions de saut-à-saut (**hbh**) et d'autre part des retransmissions de bout-en-bout (**e2e**). Dans la Figure B.9, nous représentons le côté émetteur dans la Figure B.9a et le côté récepteur dans la Figure B.9b. Les transitions en rouge correspondent au comportement des points de délestage intermédiaires alors que les autres transitions correspondent aux comportements du point de délestage source dans le cas de retransmissions de bout-en-bout (**e2e**) et

tous les points de délestage dans le cas de retransmissions de saut-à-saut (**hbh**). Le contrôleur implémente l'un ou l'autre des deux mécanismes de retransmission au niveau des points de délestage selon leur position relative dans le transfert de données via un canal de contrôle bas-débit et longue portée (*e.g.*, SigFox[5] or LoRa[6]). Chacun de ces mécanismes de retransmission repose sur un délai de temporisation (*timeout* en anglais) avant retransmission correspondant au temps de trajet moyen $t(i,j) + \varepsilon$ (ε constant) entre deux points de délestage i et j. Puisque ce temps de trajet dépend des conditions de trafic routier entre les deux points de délestage, alors le contrôleur peut utiliser des services de prévision de trafic pour prévoir et réévaluer le délai de temporisation.

Avec le premier type de retransmission de saut-à-saut (**hbh**), le contrôleur est informé des véhicules chargés quittant le point de délestage i. Une copie du cargo de données chargé sur le véhicule est mis en cache au niveau de i jusqu'à ce que ce cargo soit transmis avec succès à j. Si aucun acquittement provenant de j n'est reçu par le contrôleur avant l'expiration du délai de temporisation, alors le contrôleur notifie i de retransmettre le cargo. Cette stratégie introduit une quantité de données additionnelles à transmettre correspondant au nombre de transmissions $R_{st}^{\text{hbh}}(i,\, j)$ nécessaires pour transmettre avec succès un cargo sur $(i,\, j)$:

$$R_{st}^{\text{hbh}}(i,\, j) = \frac{1}{1 - l_{st}^{\text{red}}(i,\, j)}. \tag{B.7}$$

Avec le second type de retransmission de bout-en-bout (**hbh**), le seul point de délestage à effectuer retransmission du cargo de donnée perdu est celui placé à la source du transfert. Une fois que le dernier point de délestage placé à la destination du transfert a reçu la donnée, celui-ci notifie au contrôleur la bonne réception du cargo de données. Ensuite, le contrôleur notifie le premier point de délestage pour enlever le cargo mis en cache lors de son chargement sur le premier véhicule l'ayant transporté. Si les points de délestage intermédiaires ne reçoivent pas le cargo attendu avant le délai de temporisation, ils notifient le contrôleur qui va demander au point de délestage source de mettre à disposition une autre copie du cargo à charger sur le prochain véhicule qui s'arrête. Cette stratégie introduit une quantité de données additionnelles à transmettre correspondant au nombre de transmissions $R_{st}^{\text{e2e}}(i,\, j)$ sur le lien logique $(i,\, j)$:

$$R_{st}^{\text{e2e}}(i,\, j) = \frac{(-1)^{n-(i-1)}}{\prod_{j=i}^{n-1} \left(l_{st}^{\text{red}}(j_i,\, j_{i+1}) - 1\right)}. \tag{B.8}$$

Nous notons $O_{st}(i,\, j)$ la quantité de données additionnelles transmises par les deux mécanismes de redondance et de retransmission sur un lien logique $(i,\, j)$ pour la demande de délestage d_{st}. Celle-ci est exprimée par :

$$O_{st}(i,\, j) = \begin{cases} o_{st}^{\text{red}} \times R_{st}^{\text{hbh}}(i,j) & \text{(saut-à-saut)} \\ o_{st}^{\text{red}} \times R_{st}^{\text{e2e}}(i,j) & \text{(bout-en-bout)} \end{cases} \tag{B.9}$$

[5] `http://www.sigfox.com/`

[6] `https://www.lora-alliance.org/`

où o_{st}^{red} est donné par l'Équation B.6.

Pour la modélisation, nous utiliserons la notation $o_{st}^{\text{ret}}(i,\ j)$ pour représenter la quantité de données additionnelles transmises par le mécanisme de retransmission, qu'il soit de saut-en-saut ou de bout-en-bout. De la même manière, nous utiliserons la notation $R_{st}(i,j)$ pour représenter le nombre de transmissions sur le lien logique (i,j) d'un cargo de la demande de délestage d_{st}, indépendamment du mécanisme de retransmission utilisé.

Détermination de la taille de l'arrangement en RAID 6 (n). Pour rappel, la redondance avec RAID 6 arrange n cargos de données ensemble, incluant deux cargos de données redondants. Il nous faut déterminer le nombre de cargos dans un arrangement RAID 6 de telle manière à minimiser O_{st}, la quantité de données additionnelles à transmettre pour une demande de délestage d_{st} , en prenant en compte les cargos redondants et retransmis :

$$\min O_{st} = \begin{cases} \min\limits_{n} \left\{ o_{st}^{\text{red}} \times \max\limits_{(i,j)\in p} \{R_{st}^{\text{hbh}}(i,j)\} \right\} & \text{(saut-à-saut)} \\ \min\limits_{n} \left\{ o_{st}^{\text{red}} \times \max\limits_{(i,j)\in p} \{R_{st}^{\text{e2e}}(i,j)\} \right\} & \text{(bout-en-bout)} \end{cases}$$

B.2.4.4 Modèle max-min équitable pour l'allocation des flots de véhicules

À la réception d'une demande de délestage et de ses besoins en terme de performance, le contrôleur doit sélectionner les flots de véhicules sur les liens logiques selon les deux contraintes suivantes : (*i*) le nombre de véhicules disponibles doit être suffisant pour satisfaire aux besoins de la demande et (*ii*) l'allocation des stockages des véhicules combinés doit être efficiente et équitable entre les transferts en compétition. Le contrôleur commence par sélectionner les chemins logiques du réseau de délestage pour chacun des transferts à délester, puis alloue des flots de données sur chacun de ces chemins tout en satisfaisant les deux contraintes. Finalement, le contrôleur configure les points de délestage le long des chemins sélectionnés avec une allocation non nulle.

Dans la suite, $\mathcal{P}_{st}$ est l'ensemble des chemins logiques candidats entre s et t. Chaque chemin logique $p \in \mathcal{P}_{st}$ est une séquence de liens logiques connectant des paires de points de délestage dans le réseau de délestage. Le temps de trajet $t(p)$ d'un cargo sur le chemin logique p correspond au temps d'attente du cargo au niveau de chacun des points de délestage et de chacun des liens logiques traversés par le cargo. Ce temps de trajet est fonction de $R(i,j)$:

$$t\,(p) = \sum_{(i,\,j)\in p} R(i,\ j)\,[\delta_i + t(i,\ j)]\,, \tag{B.10}$$

où δ_i est le temps d'attente au point de délestage i.

Formulation en programme linéaire. Nous formulons le problème de l'allocation des ressources du réseau routier sous forme d'un modèle de programmation linéaire. Le modèle est présenté par l'Algorithme B.1 et consiste à allouer une quantité $f(p)$ de données aux flots de véhicules

voyageant sur les chemins logiques p de $\mathcal{P}_{st}$. Nous présentons dans un premier temps les entrées de l'algorithme d'allocation et ensuite la stratégie de l'allocation des demandes de délestage. Celle-ci repose sur un problème d'allocation de multi-flots que l'on formule à l'aide d'un programme linéaire.

Input: Ensemble des demandes de délestage $\mathcal{D} = \{d_{st}\}$, caractérisées par β_{st} et τ_{st}
Ensembles $\{\mathcal{P}_{st}\}_{st}$ de chemins logiques entre s et t pour chaque demande de $\mathcal{D}$
Temps de trajet moyen $t(p)$ sur le chemin logique p
Capacité $c(i,\, j)$ du lien logique $(i,\, j)$

Output: Allocation résultante $A = \{f(p)\}$ des flots de données associés aux chemins logiques p

Procedure Allocation **:**

$L \leftarrow \{\text{Liens logiques } (i,\, j)\}$
$\{f(p)\} \leftarrow$ Max-Min Fairness($\mathcal{D}$, L)
return $\{f(p) : p \in \mathcal{P}_{st},\, (s,\, t) \in \mathcal{D}\}$

Function Max-Min Fairness($\mathcal{D}$, L) **:**

Initialisation : $U \leftarrow \mathcal{D};\ i \leftarrow 0;\ A \leftarrow \emptyset$
while $U \neq \emptyset$ **do**
Maximiser la i-ème plus petite allocation :
$\phi_i \leftarrow$ MCF($\mathcal{D}$, C, U, i, A)
Exécuter le test de non-blocage:
$A_i \leftarrow$ Non-Blocking Test(U, A, ϕ_i)
Fixer l'allocation de chaque demande de A_i à ϕ_i
$U \leftarrow U \backslash A_i$
$i \leftarrow i + 1$
return $\{f(p) : p \in A\}$

Function MCF(D, C, U, i, A) **:**

Maximiser $\quad \phi_i$
Sujet à :

$$
\begin{array}{ll}
\sum_p f(p)\big(\tau_{st} - t(p)\big) \geq \beta_{st} & \forall (s,\, t) \in \mathcal{D} \\
\phi_i - \sum_p f(p)\big(\tau_{st} - t(p)\big) \leq 0 & \forall (s,\, t) \in U \\
\phi_k - \sum_p f(p)\big(\tau_{st} - t(p)\big) = 0 & \forall (s,\, t) \in A_k, \phi_k \text{ constant}, \\
 & k = 0,\, \ldots,\, i-1 \\
\sum_{s,\,t} o_{st}^{\text{red}} \sum_p \left[o_{st}^{\text{ret}}(i,\, j) f(p)\right] \leq c(i,\, j) & \forall (i,\, j) \in L, \\
 & p \text{ s.t. } (i,\, j) \in p
\end{array}
$$

return $\{\phi_i\} \cup \{f(p) : p \in \mathcal{P}_{st},\, (s,\, t) \in \mathcal{D}\}$

Algorithm B.1: Calcul des allocation des chemins logiques pour chaque demande de délestage en utilisant un modèle d'allocation max-min équitable.

Entrées. L'algorithme prend en entrée l'ensemble $\mathcal{D}$ de toutes les demandes de délestage. Cet ensemble inclut les demandes qui ont déjà été allouées et les nouvelles à allouer.[7] Il prend également en entrée les ensembles $\mathcal{P}_{st}$ des chemins logiques candidats entre s et t, calculés pour chacune des demandes de $\mathcal{D}$. Pour énumérer les chemins logiques de $\mathcal{P}_{st}$, nous proposons d'utiliser l'algorithme de Yen qui calcule les k plus courts chemins [Yen71] ou un algorithme de parcours de graphe en largeur [Lee61]. Dans nos simulations, nous réduisons l'espace de recherche en considérant les chemins ordonnés par temps de trajet pour un seul cargo de données. Enfin, l'algorithme prend en entrée le réseau de délestage et les propriétés de chacun des liens logiques (*i.e.*, capacité, temps de trajet et taux de perte).

Modèle d'allocation max-min équitable. Le contrôleur alloue un quantité $f(p)$ de données à chacun des chemins logiques p de $\mathcal{P}_{st}$ selon la stratégie max-min équitable. Cette stratégie procède par itérations successives. La première itération alloue la quantité minimale de flots de données de manière à satisfaire les besoins d'au moins une demande (donnés par la première contrainte de la fonction MCF). Les itérations successives allouent les capacités restantes des liens logiques du réseau de délestage aux demandes qui peuvent recevoir une plus grande quantité de flot. Plus précisément, l'itération i maximise l'allocation du flot minimal noté ϕ_i et fixe l'allocation pour les demandes qui ne peuvent pas être mieux servies (*i.e.*, soit du fait des contraintes de capacité sur les liens logiques, soit parce que les besoins des demandes ont été satisfaits par l'allocation fixée). Nous utilisons l'algorithme de test de non-blocage de l'Algorithme B.2 afin de déterminer pour quels transferts de l'itération courante l'allocation peut être augmentée à l'itération suivante.

La fonction MCF (*Multi-Commodity Flow*) est au cœur de l'algorithme max-min équitable et est définie dans l'Algorithme B.1. Celle-ci calcule une allocation de multi-flots. La première contrainte fait correspondre la quantité de données qui peut être délestée pendant le temps imparti τ_{st} avec la quantité de données à transférer β_{st}. La contrainte suivante garantie que les demandes appartenant aux ensembles A_k, $k \in [0,\, i-1]$ sont fixées à l'allocation qu'elles ont reçue à l'itération k. L'objectif de la fonction MCF est de maximiser l'allocation minimum ϕ_i telle que toutes les demandes soient satisfaites. Cet objectif est garanti par la troisième contrainte du problème linéaire. Enfin, la dernière contrainte limite la quantité cumulée des allocations sur les chemins logiques à la capacité maximale des liens logiques. Cette contrainte prend en compte les données additionnelles transmises avec les mécanismes de redondance et de retransmission.

Une fois que l'allocation ϕ_i a été fixée par la fonction MCF à l'itération i, l'algorithme max-min équitable détermine quelles demandes peuvent être augmentées par rapport à leur allocation courante en utilisant le test de non-blocage de l'Algorithme B.2. Le test de non-blocage est adapté de l'algorithme proposé par Pióro *et al.* [PNKF03]. Ce test compare le débit maximal des flots alloués par une allocation multi-flots par rapport à celle qui résulte de l'allocation minimale ϕ_i. Si l'allocation multi-flots augmente la quantité maximale de données allouée à une demande, celle-ci est non-bloquante et peut être fixée dans l'une des prochaines itérations de

[7] Il nous faut réallouer les demandes déjà allouées pour assurer de l'équité entre ces demandes et les nouvelles.

Input: Ensemble U de demandes allouées à l'itération i
Ensemble A de demandes allouées aux itérations $k < i$
Allocation ϕ_i de l'itération i
Output: Ensemble A_i des flots de U qui ne peuvent pas recevoir une allocation supérieure à ϕ_i

Function Non-Blocking Test($U,\ A,\ \phi_i$) **:**

Les demandes qui ont été satisfaites sont allouées:
$D_i = \{(s,\, t)$ t.q. $\sum_p f(p)\big(\tau_{st} - t(p)\big) = \beta_{st}\}$
$U \leftarrow U \backslash D_i;\ A_i = D_i$
Les demandes allouées plus que ϕ_i peuvent recevoir une allocation supérieure :
$U_i \leftarrow \{(s,\, t)$ t.q. $\sum_p f(p)\big(\tau_{st} - t(p)\big) > \phi_i\}$
Test de l'allocation des demandes restantes :
foreach demande $(s,\, t) \in U \backslash (A_i \cup U_i)$ **do**
Résoudre le programme linéaire suivant :

Maximiser	$\sum_{p \in \mathcal{P}_{st}} f(p)$
Sujet à:	
$\sum_p f(p)\big(\tau_{st} - t(p)\big) \geq \beta_{st}$	$\forall (s,\, t) \in D$
$\phi_k - \sum_p f(p)\big(\tau_{st} - t(p)\big) = 0$	$\forall (s,\, t) \in A_k, \phi_k$ constant, $k = 0,\, \ldots,\, i-1$
$\sum_{s,\,t} o_{st}^{\text{red}} \sum_p \left[o_{st}^{\text{ret}}(i,\, j) f(p)\right] \leq c(i,\, j)$	$\forall (i,\, j) \in C$, p s.t. $(i,\, j) \in p$

if $\sum_{p \in \mathcal{P}_{st}} f(p) \leq \phi_i$ **then** $A_i \leftarrow A_i \cup \{(s,\, t)\}$

else $U_i \leftarrow U_i \cup \{(s,\, t)\}$

return A_i

Algorithm B.2: Test de non-blocage pour le modèle d'allocation max-min équitable.

l'algorithme max-min équitable. Sinon, l'allocation de la demande ne peut pas être augmentée et est fixée à ϕ_i.

Cette formulation d'allocation multi-flots reste valide pour des flots continus de données (par exemple, des demandes avec un débit constant). Dans ce cas, il nous suffit d'adapter la première contrainte de la fonction MCF pour satisfaire aux besoins de la demande en l'exprimant en fonction du débit à atteindre.

B.2.4.5 Ordonnancement des données

Afin d'ordonnancer les données, chaque point de délestage utilise des "tables de flots" configurées par le contrôleur suite à l'allocation des flots de véhicules pour faire correspondre les données disponibles localement avec les véhicules à l'arrêt et décider d'une action à prendre (*i.e.*, charger un cargo de données sur le véhicule, décharger un cargo ou échanger des cargos). Une entrée de la table de flot contient le prochain saut (*i.e.*, point de délestage) correspondant

au chemin logique alloué pour une demande de délestage donnée. La direction du véhicule à l'arrêt est prédite par le contrôleur et mise en correspondance avec le prochain saut de chacune des entrées de la table de flot (pour chacune des entrées dont les cargos de données associés sont disponibles localement au point de délestage). Si aucune des entrées ne correspond à la destination du véhicule, alors le cargo transporté par le véhicule est déchargé et mis en transbordement au niveau du point de délestage pour être transporté par les véhicules suivants. Dans le cas où plusieurs entrées correspondent à la direction du véhicule, le point de délestage sélectionne l'entrée selon la politique d'ordonnancement décrite ci-après. Si le véhicule transporte un cargo de données, le point de délestage vérifie qu'elles n'appartiennent pas déjà au flot sélectionné. Si tel est le cas, aucune action n'est entreprise et le véhicule continue sa route. Sinon, le cargo est déchargé du véhicule. Dans le cas d'un véhicule vide, le cargo le plus ancien de données correspondant aux entrées sélectionnées est chargé sur le véhicule.

Dans le cas où plusieurs entrées de la table de flot du point de délestage i correspondent à la direction du véhicule à l'arrêt, i sélectionne l'entrée selon une politique d'ordonnancement de type *weigthed fair queuing*, dont les poids sont configurés par le contrôleur suivant le résultat de l'allocation des flots de véhicules. Nous notons $f(d_k p_l)$ le flot de données sur le chemin logique p_l alloué au transfert résultant de la demande de délestage d_k. Le contrôleur assigne un poids $w(d_k p_l)$ à chacun des transferts alloués sur le chemin logique $P = \{d_k p_l | \forall d_k \in D,\, p_l \in \mathcal{P}^{d_k},\, (i,\, j) \in p_l\}$, où P est l'ensemble des chemins logiques traversant à la fois le point de délestage i et le lien logique $(i,\, j)$ connectant i au prochain saut j. Le contrôleur calcule le poids $w(d_k p_l)$ en normalisant le débit du flot $f(d_k p_l)$ par rapport au débit total des flots de P :

$$w(d_k p_l) = \frac{f(d_k p_l)}{\sum_{p \in P} f(p)}. \tag{B.11}$$

Nous utilisons les poids de l'algorithme d'ordonnancement pour déterminer la priorité des transferts de données à assigner aux cargos de données pour en charger un dans le véhicule à l'arrêt, dans le cas où plusieurs allocations de transferts traversent le même point de délestage. Dans nos simulations, nous considérons une politique d'ordonnancement de type *weigthed fair queuing* qui utilise ces poids afin de sélectionner de manière aléatoire un cargo de données à charger dans le véhicule à l'arrêt [SV96].

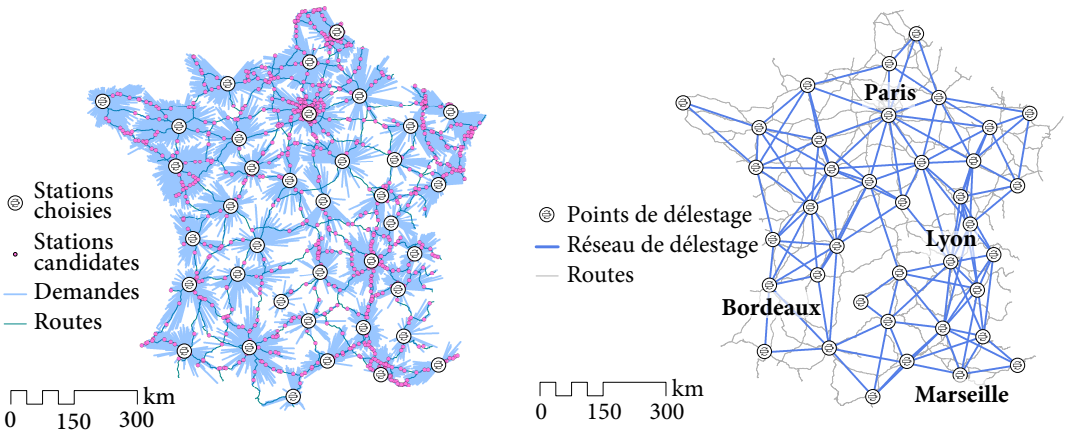
B.2.5. Évaluation sur les routes françaises

Jeu de données. Pour évaluer le délestage des données, nous utilisons un jeu de données routières de la DTITM (Direction Technique Infrastructures de Transports et Matériaux). Ces données comportent les trafics moyens journaliers annuels (TMJA) de 2011 sur 20 000 km d'autoroutes et routes nationales en France.[8]

Pour les besoins de cette évaluation, nous avons simulé un plan de déploiement de ces stations sur le territoire français à l'aide d'un algorithme d'allocation de facilités. Les stations sont allouées pour maximiser les demandes venant des villes, pondérées par leur taille, de

[8] Données de la DGITM/DIT – SETRA – IGN (`http://tinyurl.com/otfbewv`)

telles sorte qu'elles soient toutes placées au plus à 150 km l'une de l'autre (représenté sur la Figure B.10a). En effet, l'autonomie moyenne d'un véhicule étant de 300 km, cette allocation permet à un véhicule de rejoindre la prochaine station sur sa route ou de rejoindre la précédente. L'allocation donne 38 stations de chargement réparties sur le réseau routier français, comme montré sur les Figures B.10a et B.10b.



(a) Allocation des stations de chargement de batteries électriques sur le réseau routier français. Les gros points sont les stations choisies et les lignes représentent les demandes allouées aux stations choisies.

(b) Le réseau de délestage construit à partir du réseau routier français. Les points de délestage sont ceux représentés sur la Figure B.10a.

Figure B.10. Résultat de l'allocation de facilité et réseau de délestage. Les gros points sont les stations de chargement choisies.

Création du réseau de délestage. Le réseau de délestage est créé en considérant les stations de chargement précédemment déployées. Le but est de caractériser ses liens logiques par des quantités réseau en déterminant les volumes de trafic routier entre chaque paire de points de délestage. Ces volumes sont donnés par des matrices origine-destination connectant les points de délestage. Cependant, ces matrices sont rarement disponibles (voire pas du tout disponibles) pour des réseaux routiers à large-échelle. Nous les déduisons de jeux de données disponibles publiquement contenant les valeurs de Trafic Moyen Journalier Annuel (TMJA) pour chacun des tronçons de route du réseau routier français. Ainsi, en utilisant des techniques bien connues de prévision de trafic routier issues des recherches en transportation, nous sommes à même d'en déduire les matrices origine-destination nécessaire à notre caractérisation du réseau de délestage. Nous procédons aux trois étapes suivantes :

1. *Détermination des routes.* La première étape consiste à sélectionner un sous-ensemble de routes alternatives (à la route la plus courte) du réseau routier connectant chaque paire de points de délestage. Cette étape consiste à sélectionner les k routes les plus courtes en terme de temps de trajet. Les routes sont sélectionnées de telle manière qu'elles ne partagent qu'un faible degré de similarité entre elles en terme de tronçons de route en commun. Nous implémentons cette étape en utilisant les algorithmes proposés par Abraham *et al.* [ADGW13].

2. *Assignement des routes.* La seconde étape consiste à pondérer les routes sélectionnées en utilisant le modèle d'assignement des routes C-logit [CN96]. Les valeurs de poids sont déterminées selon le temps de trajet et la distance des routes. Ces poids reflètent la capacité d'une route à attirer du trafic, avec une grande valeur de pondération, la route attire plus de trafic routier par rapport à une route avec une petite valeur de pondération. Les poids sont ensuite utilisés avec les TMJA des tronçons pour estimer les volumes de trafic entre les points de délestage.
3. *Estimation de la matrice origine-destination.* Dans cette troisième étape, nous utilisons un modèle de maximisation d'entropie proposé par Zuylen et Willumsen pour calculer la matrice d'origine-destination entre chaque paire de points de délestage du réseau de délestage [VZW80]. Ce modèle détermine la distribution de trafic la plus probable sur les routes déterminées durant la première étape et est sujet à deux contraintes : (*i*) la capacité des tronçons de route pour les routes sélectionnées données par les TMJA et (*ii*) les poids déterminés par le modèle d'assignement des routes C-logit présenté lors de la deuxième étape. Nous utilisons un jeu de données donnant les TMJA des tronçons de route de France afin d'en inférer les matrices origine-destination.

Finalement, à partir de la matrice origine-destination entre chaque paire de point de délestage, nous déterminons les attributs de chaque lien logique (i, j) du réseau de délestage. Pour rappel, ces attributs sont la capacité, le temps de trajet et le taux de perte. La Figure B.11 montre le nombre de chemins en fonction du temps de trajet. Dans chacun des deux cas, le nombre de chemins possibles augmente de manière exponentielle avec le temps de trajet. Cependant, le nombre de chemins est beaucoup plus grand pour le réseau routier, comparé au nombre de chemins du réseau de délestage. De plus, la différence du nombre de chemins possibles augmente de manière exponentielle également. C'est cette différence qui consititue le facteur principal de complexité quand il s'agit de résoudre le problème de l'allocation de flots de véhicules. Ainsi, ce réseau de délestage permet de simplifier l'infrastructure routière tout en préservant les propriétés des chemins acceptables avant agrégation.

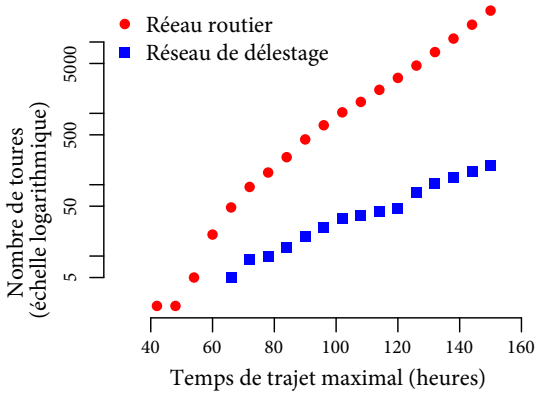


Figure B.11. Nombre total de chemins dans le réseau routier français et du réseau de délestage en fonction du temps de trajet maximal.

Modèle d'allocation max-min équitable sur le réseau routier de France. Nous évaluons les performances des transferts résultant de l'allocation de *trois* demandes de délestage. Celles-ci sont représentées sur la Figure B.12 : (*i*) d_A de Paris à Lyon avec un taux d'arrivées Poissoniennes λ_A, (*ii*) d_B de Paris à Bordeaux avec un taux d'arrivées Poissoniennes λ_B, et (*iii*) d_C de Paris à Marseille avec un taux d'arrivées Poissoniennes λ_C. Il faut noter que les chemins suivis par les transferts résultant des demandes d_A et d_C partagent le même lien logique dans le réseau de délestage. De fait, d_A et d_C vont être en compétition pour prendre la capacité de ces liens logiques.

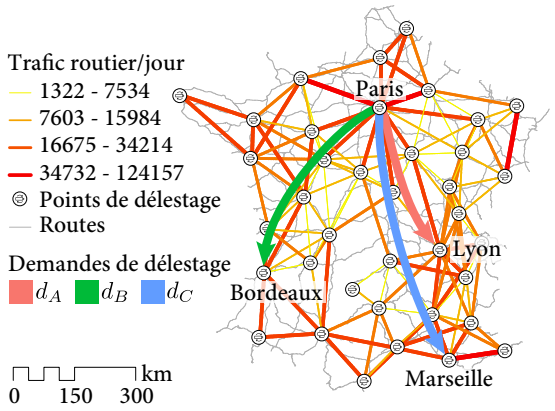


Figure B.12. Représentation schématiques des trois demandes de délestage d_A, d_B et d_C.

Nous choisissons des paramètres volontairement pessimistes: nous supposons uniquement 10% du parc automobile français équipé de capacités de stockage égal à 1 Téraoctet sur chaque véhicule. Nous supposons que les stations de chargement ont une capacité qui correspond à la demande pour charger des véhicules électriques, ce qui donne un temps d'attente nul avant d'être servi. Le temps d'attente est donc réduit au temps de service, qui correspond au temps de charge de la batterie électrique. Nous supposons le taux d'erreur sur les liens logiques du réseau de délestage égal pour chacun des liens, indépendamment de leur longueur ou volume de trafic.

Nous effectuons les simulations avec le simulateur de trafic SUMO (*Simulation of Urban MObility*), duquel nous récupérons et traitons les évènements via l'API TraCI [BBEK11]. La durée de chaque simulation est de 300 000 secondes (3,5 jours), incluant 43,200 secondes (12 heures) d'initialisation (*warmup* en anglais) pour remplir les stockages des points de délestage.

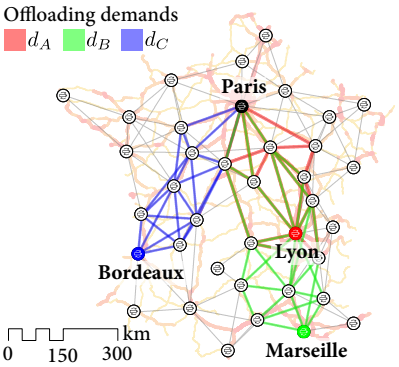


Figure B.13. Chemins logiques alloués aux demandes d_A, d_B et d_C.

Débit maximum atteignable. Nous évaluons le débit maximum atteint par chacun des transferts d_A, d_B et d_C résultant de la stratégie d'allocation max-min équitable présentée dans la Section B.2.4.4. Nous considérons également une autre stratégie sans contrôleur identique à celle présentée dans la section B.2.3 qui sélectionne les cargos de données à charger dans les véhicules à l'arrêt selon la politique d'ordonnancement *Round-Robin*. Nous considérons un trafic ininterrompu de données généré par une seule source située à Paris pour chacun des transferts (*i.e.*, $\lambda_A = \lambda_B = \lambda_C = \infty$). Nous évaluons le débit maximum atteint pour chaque stratégie et chaque mécanisme de retransmission (*i.e.*, saut-à-saut et bout-en-bout) présentés dans la Section B.2.4.3. Celui- est représenté sur la Figure B.14 en fonction du taux d'erreur sur les liens logiques. Le débit est exprimé en terme de nombre de cargos de données livrés par seconde.

Dans un premier temps, nous examinons le débit résultant de la stratégie sans contrôleur. Cette stratégie ne garantie pas un débit réparti équitablement entre les transferts résultant des trois demandes. Le débit maximum pour la demande d_C est inférieur comparé à ceux des deux demandes d_A et d_B. Comme montré sur la Figure B.13 (dans le cas avec contrôleur), les transferts de données résultants des demandes d_A et d_C sont en compétition pour les mêmes ressources, puisqu'elles partagent toutes deux des liens logiques. La stratégie d'ordonnancement sans contrôleur alloue les flots de véhicules voyageant le long de ces liens vers les destinations respectives de d_A et d_C sans prendre en compte le fait que la destination de d_C est plus éloignée que celle

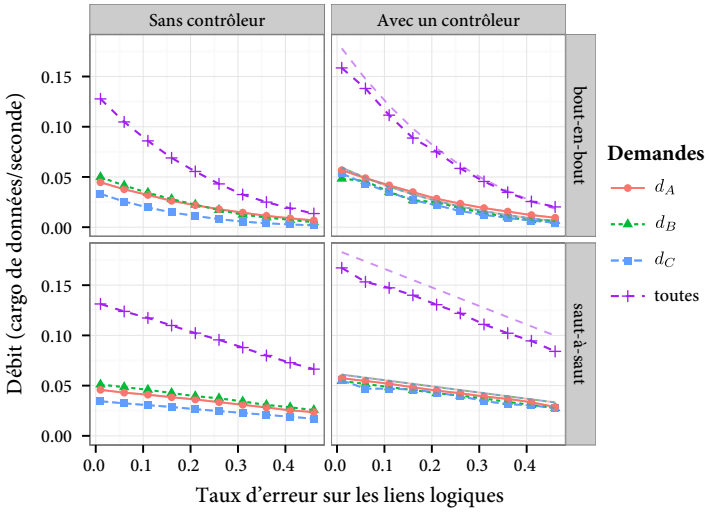


Figure B.14. Débit maximum pour les demandes de délestage d_A, d_B et d_C (représentées dans la Figure B.12) en fonction du taux d'erreur sur les liens logiques (supposé égal pour chacun des liens logiques).

de d_A. Ainsi, la stratégie favorise d_A au dépend de la demande d_C. Le transfert résultant de la demande d_B, quant à lui, n'est pas affecté puisque les flots de véhicules alloués à d_B voyagent le long de chemins logiques différents par rapport à d_A et d_C. Nous pouvons aussi noter que le débit maximum pour les transferts d_A et d_B ont des valeurs similaires puisque leur destinations sont équidistantes de leur source.

Dans un second temps, nous examinons la stratégie avec un contrôleur. Nous constatons que cette stratégie donne de meilleurs débits cumulés que la stratégie sans contrôleur. C'est la conséquence directe de la conception des deux stratégies d'ordonnancement. La stratégie avec contrôleur utilise une allocation max-min équitable qui alloue les flots de véhicules de façon à maximiser les flots de données alloués et à garantir l'équité entre les transferts alloués en concurrence. L'allocation alloue les chemins logiques pour chacune des demandes comme représenté sur la Figure B.13. La meilleure performance de cette stratégie confirme le besoin d'un contrôleur, comme nous l'avions montré dans la Section B.2.3. Les résultats confirment également que la stratégie avec un contrôleur garantie une allocation équitable entre les transferts résultants des trois demandes de délestage.

Enfin, le mécanisme de retransmission de type saut-à-saut donne de meilleurs résultats par rapport à celui de type de bout-en-bout. Le premier mécanisme recouvre les erreurs plus rapidement que le second, qui demande à la source de retransmettre le cargo de données. Celui-ci doit alors parcourir à nouveau tout le chemin déjà parcouru par le cargo perdu. En contrepartie, la stratégie de type bout-en-bout nécessite moins de stockage localement au niveau des points de délestage pour mettre en cache les cargos chargés dans les véhicules (les cargos mis en cache sont utilisés par la stratégie de type saut-à-saut pour récupérer les cargos perdus).

Pour des cargos de taille 1 Téraoctet, l'allocation qui résulte de la stratégie avec un contrôleur

donne un débit cumulé de 114 Gigabits par seconde en utilisant le système de retransmission de type saut-à-saut et avec un taux d'erreur de 30%. Cela correspond à une moyenne de 38 Gigabits par seconde pour chacun des trois transferts.

B.3 | Extension à des services véhiculaires de type "cloud"

Dans ce chapitre, nous étendons le concept de point de délestage selon deux directions afin d'en faire une base pour développer des services véhiculaires de type "cloud".

Dans une première section, nous exploitons les capacités de stockage des points de délestage pour concevoir un système distribué de stockage et de partage de fichiers générés par des utilisateurs mobiles. Les points de délestage jouent le rôle de dépôts où les utilisateurs mobile y déposent et récupèrent des fichiers. Pour augmenter les chances de trouver le fichier recherché en un temps imparti, les dépôts distribuent les fichiers entre eux en utilisant les mouvements existants des utilisateurs les visitant.

Dans une seconde partie, nous dématérialisons les points de délestage en zones prédéfinies où les véhicules sont en contact fréquemment et suffisamment longtemps pour pouvoir transférer des fichiers de grosse taille. Nous utilisons ces zones dans le contexte de virtualisation de réseaux véhiculaires où les ressources des véhicules (*i.e.*, calcul, stockage et communications) sont partagées en utilisant des machines virtuelles. Les machines virtuelles sont allouées à un fournisseur de service qui utilise les mouvements des véhicules hébergeant les machines virtuelles pour déployer des services géo-distribués à large-échelle (par exemple, des plateformes de capteurs). En particulier, nous nous intéressons à la capacité des contacts entre les véhicules dans ces zones prédéfinies pour effectuer les migrations de machines virtuelles nécessaires en cas de changements de topologie ou de réallocation des ressources des véhicules.

Dans chacune de ces deux extensions, nous proposons un placement optimisé des points de délestage selon les besoins spécifiques aux services à mettre en place. Ainsi, nous utilisons les concepts vus dans la première partie de cette thèse avec le réseau de délestage pour obtenir une vue logique des mouvements des véhicules entre les points de délestage.

B.3.1. Système véhiculaire de stockage et partage de fichiers

Dans cette section, nous présentons un système de stockage et de partage de fichiers pour utilisateurs mobiles. Ce système repose sur un ensemble de facilités dédiées équipées de stockage pré-positionnées à des endroits stratégiques qui jouent le rôle de dépôts pour fichiers. Les utilisateurs mobiles stockent leurs fichiers qu'ils souhaitent archiver ou partager avec d'autres utilisateurs, puisque ces fichiers peuvent être récupérés plus tard par leur propriétaire ou par d'autres utilisateurs depuis n'importe quel dépôt du système. Les utilisateurs émettent des requêtes pour stocker et récupérer des fichiers dans les dépôts du système. Afin de limiter le nombre de requêtes actives simultanément, celles-ci disposent d'un temps imparti avant lequel elles doivent être satisfaites. Au bout du temps imparti, la requête expire et est considérée comme étant non satisfaite.

Le système ne considère aucune connexion réseau disponible entre les dépôts, rendant impossible de récupérer un fichier depuis un dépôt autre que celui où le fichier avait été stocké en premier lieu. Afin de résoudre ce problème, le système exploite les mouvements des utilisateurs entre les dépôts pour partager les copies des fichiers stockés et ainsi les répliquer au sein des dépôts du système. Notons qu'exploiter la mobilité des utilisateurs limite la dépendance vis-à-vis des réseaux de données traditionnels qui pourraient être utilisés pour connecter les dépôts entre eux. Afin d'augmenter les chances de répliquer les fichiers entre les dépôts, plusieurs copies des fichiers sont transportés par différents utilisateurs.

Le problème qui nous intéresse dans ce travail est de placer un nombre cible de dépôts en fonction du temps imparti pour satisfaire les requêtes des utilisateurs. De ce fait, le placement est contraint par les deux objectifs suivants :

1. Déterminer les emplacements des dépôts pour qu'ils servent un nombre maximal de requêtes avant qu'elles n'expirent.
2. Connecter les dépôts entre eux par les mouvements des utilisateurs afin de répliquer les fichiers entre dépôts.

B.3.1.1 Placement des dépôts

Les différentes étapes de l'algorithme du placement des dépôts sont illustrées avec la Figure B.15 dans le cas d'un système de bus. La couche inférieure montre les trajectoires de bus du quartier des affaires de San Francisco aux États-Unis et le sous-ensemble d'arrêts de bus jouant le rôle de dépôts de fichiers. Chaque ligne de bus est représentée par une couleur et une épaisseur indiquant la fréquence à laquelle les bus voyagent sur cette ligne. La couche du milieu montre les demandes discrétisées représentant les requêtes générées par les passagers de bus. De plus, cette couche montre comment les demandes discrétisées sont allouées aux différents dépôts selon leur distance respective aux dépôts. Enfin, la couche supérieure montre un graphe logique dont les nœuds correspondent aux dépôts et les liens correspondent aux flots de bus voyageant entre les dépôts. De la même manière que pour la couche inférieure, l'épaisseur des liens correspond à la fréquence des routes des bus.

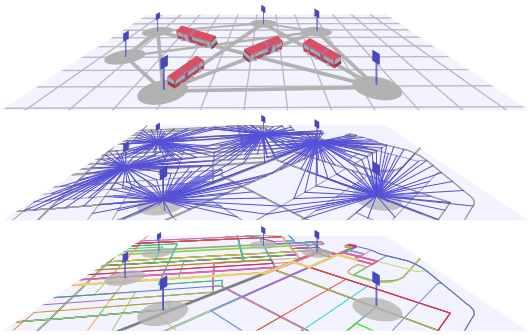


Figure B.15. Différentes couches illustrant les différentes étapes de l'algorithme de placement des dépôts dans le cas d'un système de bus.

La formulation du placement des dépôts est dérivée du problème de couverture par ensembles, et en particulier le problème de couverture maximale [CV74]. Étant donné un ensemble d'emplacements candidats pour des dépôts, le problème de placement consiste à sélectionner les dépôts tels que leur couverture maximise le taux de succès des requêtes générées par les utilisateurs. Ce problème a été montré NP-Complet [MZH83], donc il nous faut adapter des

heuristiques connues pour le résoudre. De ce fait, nous considérons l'heuristique gloutonne (*greedy* en anglais) d'ajouts avec substitutions (GAS) [CV74] qui détermine les emplacements optimaux pour chaque itération de l'algorithme.

B.3.1.2 Simulation avec des traces réalistes

Afin d'évaluer l'algorithme de placement de dépôts, nous effectuons dans un premier temps un déploiement d'un nombre prédéfini de dépôts en considérant les arrêts et routes des bus opérés par MUNI à San Francisco aux États-Unis.[9] Nous recréons les mouvements des bus et ensuite sélectionnons les arrêts où déployer les dépôts.

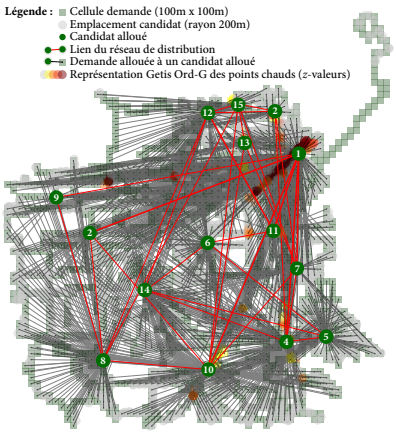
Nous faisons les hypothèses suivantes. Les utilisateurs mobiles sont des passagers des bus. Les dépôts sont équipés d'interfaces réseau de type IEEE 802.11 avec une portée de 250 m. Pour un souci de simplicité, nous ignorons les interférences et supposons que la capacité de ces interfaces est suffisante pour transférer l'ensemble des fichiers (quel qu'il soit) entre les dépôts et les passagers des bus. Les bus s'arrêtent aux arrêts pour une durée aléatoire comprise entre 10 secondes et 30 secondes. Par ailleurs, nous ne faisons aucune supposition sur la popularité géographique des fichiers : une requête est générée par un utilisateur en moyenne chaque 10 minutes de temps simulé selon une distribution Poissonienne. Nous considérons deux types de requêtes avec les proportions suivantes : 10% des requêtes sont des requêtes pour stocker un fichier et 90% des requêtes récupèrent un fichier. La durée de simulation est de 20 000 secondes.

Recréer les mouvements des bus. Nous avons développé un logiciel pour analyser les données de mobilité à partir de traces de véhicules réelles. Le jeu de données que nous utilisons décrit les mouvements prévisionnels des bus selon le format standard GTFS (*General Transit Feed Specification*).[10] Ce logiciel permet également d'inférer les mouvements complets des bus à partir de jeux de données de type GTFS. Le format des mouvements inférés est compatible avec le simulateur à évènements discrets ONE [KOK09]. À partir des traces GTFS des bus de San Francisco, nous avons recréé les mouvements de 493 bus sur 130 routes pour des trajets faits entre 10h et 16h durant un jour représentatif de la moyenne. Nous utilisons alors le simulateur ONE pour simuler les mouvements des bus et leurs connexions réseau avec les dépôts.

Placement des dépôts dans la ville de San Francisco. Les dépôts sont placés à l'aide de l'algorithme GAS itératif décrit dans la Section B.3.1. Nous simulons les demandes représentant les requêtes émises par les passagers des bus en discrétisant l'espace géographique en cellules carrées de taille 100 m $\times$ 100 m, chacune représentée par un poids égal au ratio des visites des bus dans les cellules et de la durée entre deux visites consécutives à partir de chacun des candidats. À chaque itération de l'algorithme, un arrêt de bus est choisi parmi ceux candidats pour faire office de dépôt de telle façon à maximiser le poids de chacune des demandes situées à la périphérie de l'arrêt (*i.e.*, celle dont le temps de trajet médian est inférieur au temps imparti donné). Si

[9] `https://www.sfmta.com/`

[10] `https://developers.google.com/transit/gtfs`



(a) Placement des dépôts. Les variations de couleurs des demandes et des candidats représentent des points chauds (*hotspot* en anglais), correspondant à des emplacements signifiants spatialement, calculé en utilisant les p-valeurs des fréquences moyennes de visites de l'analyse spatiale Getis Ord-G [GO92].

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	51 (7)	9 (3)	28 (5)	35 (8)	54 (12)	52 (9)	31 (6)	40 (5)	87 (13)	49 (5)	34 (7)	6 (3)	9 (3)	47 (8)	0 (0)
14	55 (7)	51 (8)	63 (9)	32 (12)	36 (10)	69 (12)	40 (12)	63 (12)	108 (17)	40 (12)	48 (9)	41 (9)	57 (9)	0 (0)	49 (10)
13	50 (6)	21 (5)	39 (6)	42 (5)	58 (10)	50 (7)	35 (8)	52 (6)	96 (13)	44 (6)	31 (5)	18 (5)	0 (0)	59 (9)	10 (3)
12	46 (5)	15 (4)	23 (5)	38 (6)	56 (12)	46 (9)	37 (5)	35 (4)	81 (14)	54 (4)	27 (7)	0 (0)	16 (5)	41 (10)	7 (3)
11	19 (3)	40 (6)	49 (5)	17 (4)	31 (9)	19 (6)	25 (6)	62 (6)	84 (18)	48 (7)	0 (0)	27 (6)	33 (7)	47 (10)	33 (6)
10	31 (4)	52 (12)	40 (8)	45 (15)	59 (19)	61 (6)	30 (17)	24 (8)	69 (12)	0 (0)	47 (7)	54 (4)	45 (6)	41 (13)	50 (5)
9	70 (20)	95 (11)	61 (13)	92 (19)	108 (19)	96 (20)	83 (16)	44 (14)	0 (0)	71 (14)	84 (17)	84 (14)	98 (14)	108 (17)	90 (13)
8	52 (6)	50 (7)	14 (4)	64 (10)	80 (15)	78 (7)	53 (13)	0 (0)	46 (14)	25 (8)	59 (5)	34 (4)	50 (6)	63 (12)	41 (6)
7	28 (12)	23 (8)	54 (6)	9 (7)	27 (11)	44 (7)	0 (0)	59 (14)	89 (18)	32 (17)	26 (6)	38 (6)	38 (9)	39 (12)	35 (9)
6	33 (4)	61 (8)	67 (6)	36 (7)	39 (8)	0 (0)	43 (7)	80 (6)	99 (19)	62 (6)	19 (6)	47 (8)	53 (10)	67 (12)	55 (8)
5	44 (12)	43 (11)	72 (9)	12 (9)	0 (0)	41 (12)	23 (10)	75 (12)	104 (20)	50 (14)	32 (11)	54 (10)	54 (9)	36 (12)	52 (8)
4	28 (6)	28 (9)	59 (6)	0 (0)	14 (8)	36 (7)	9 (6)	65 (12)	94 (19)	40 (15)	16 (4)	40 (6)	42 (6)	30 (12)	39 (6)
3	41 (8)	39 (6)	0 (0)	59 (6)	79 (12)	67 (7)	55 (6)	16 (3)	64 (16)	41 (8)	49 (6)	23 (4)	39 (6)	63 (8)	30 (6)
2	50 (11)	0 (0)	40 (6)	26 (9)	45 (12)	58 (8)	24 (9)	53 (6)	96 (14)	51 (12)	40 (6)	17 (5)	21 (5)	50 (7)	12 (4)
1	0 (0)	49 (11)	40 (8)	27 (5)	44 (11)	32 (5)	28 (12)	52 (6)	68 (20)	31 (4)	17 (3)	45 (8)	50 (8)	60 (12)	50 (7)

(b) Matrice origine-destination des dépôts montrant le temps moyen de trajet entre chaque paire de dépôts (identifiés par les numéros représentés sur la Figure B.16a). Les valeurs entre parenthèses sont les écart-types correspondants.

Figure B.16. Résultats de l'allocation de 15 dépôts à San Francisco en utilisant les mouvements des bus du système MUNI.

plusieurs dépôts sont déjà alloués, alors l'ensemble des candidats est réduit à ceux qui sont connectés aux candidats déjà alloués.

Nous représentons le résultat du placement sur la Figure B.16a. La figure est le résultat du placement de 15 dépôt parmi les 4 590 arrêts de bus de San Francisco. Sur cette figure, nous présentons une carte des points chauds pour chacune des cellules où les teintes les plus foncées représentent un poids agrégé plus élevé. Ces cartes de points chauds nous aident à représenter et à analyser concrètement les données, afin de raffiner le placement des dépôts.

Durée de la distribution. La première métrique de performance que nous évaluons est le temps nécessaire pour distribuer un fichier dans tous les dépôts. La Figure B.17 montre la durée moyenne pour distribuer un fichier en fonction de la disponibilité des fichiers dans le système. Celle-ci est mesurée à partir de l'instant où le fichier est stocké dans le premier dépôt, jusqu'à l'instant où il est distribué à tous les autres dépôts. À noter que le premier dépôt où le fichier est stocké peut être n'importe quel dépôt disponible.

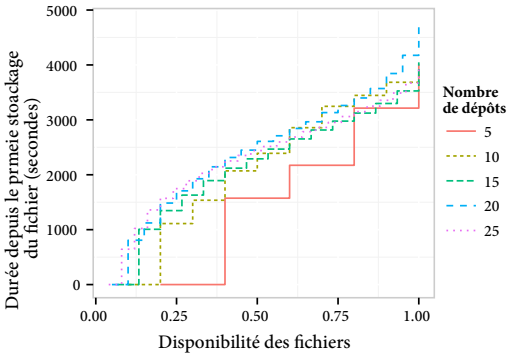


Figure B.17. Durée de la distribution en fonction de la disponibilité des fichiers dans le système pour différents nombres de dépôts.

La durée moyenne pour distribuer un fichier, quelque soit le nombre de dépôts disponibles est

de environ 4 000 secondes, soit un peu plus d'une heure. Cela prend plus de temps pour distribuer les fichiers au début, avec une seule copie disponible. Avec plus de dépôts qui ont une copie, plus de copies du fichier sont distribuées et la disponibilité augmente plus rapidement. Enfin, notons que cela prend plus de temps pour atteindre le dernier dépôt, puisque c'est généralement le plus éloigné du dépôt de la première copie.

Sur la Figure B.16b, nous donnons une matrice origine-destination montrant le temps de trajet moyen en minutes entre n'importe quelle paire de dépôts (le nombre de sauts empruntés par le fichier n'est pas représenté). Cela se traduit par la durée moyenne pour distribuer un fichier une fois le fichier stocké dans un dépôt (ce temps prend également en compte les attentes entre deux bus successifs s'il y a un ou plusieurs sauts intermédiaires). Notons que la connectivité des dépôts entre eux dépend du nombre et de la fréquence des trajets en bus les reliant. À titre d'exemple, les dépôts 1 et 15 sont très bien connectés entre eux et disposent d'un temps de trajet moyen inférieur à la moyenne. Ce n'est pas le cas du dépôt 9 qui est plus éloigné du reste, ce qui explique pourquoi cela prend plus longtemps que la moyenne pour y distribuer des fichiers.

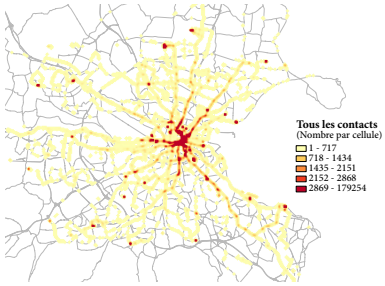
B.3.2. Migration de machines virtuelles dans des réseaux véhiculaires

Dans cette section, nous présentons le problème de la virtualisation des ressources véhiculaires dans le cadre d'un réseau mobile large-échelle. En particulier, nous nous intéressons à la migration de machines virtuelles déclenchées par des réallocations des ressources virtuelles ou des changements dans la topologie physique (*e.g.*, un véhicule ne participe plus à la virtualisation). Afin d'exécuter la phase de migration, au lieu d'utiliser les connexions cellulaires de type 3G ou 4G, nous proposons d'utiliser les communications de véhicule-à-véhicule. Ainsi, nous étudions la caractérisation des opportunités de "contacts" entre bus (*i.e.*, lorsque deux bus entrent tous deux dans la portée de communication de chacun) dans un contexte urbain large-échelle.

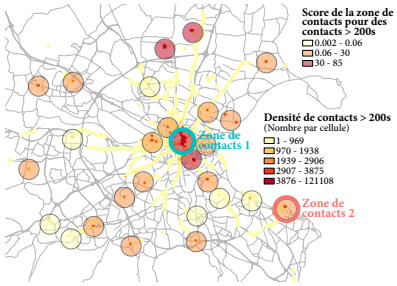
Ce travail s'inscrit dans les résultats présentés pour des réseaux similaires de différentes villes, par exemple San Francisco aux États-Unis et Varsovie en Pologne, où les auteurs ont montré que les véhicules entrent en contact plus souvent et pour des durées plus longues à des endroits bien précis [SDPG06]. Ainsi, en limitant les migrations de machines virtuelles à ces endroits, ou "zones", nous espérons une amélioration des migrations de machines virtuelles. Nous présentons (*i*) une analyse des opportunités de contacts entre bus de la ville de Dublin en Irlande et (*ii*) une méthodologie similaire à celle présentée dans la Section précédente B.3.1.2 pour déterminer ces zones et évaluer les opportunités de migration de machines virtuelles via les communications bus-à-bus.

B.3.2.1 Génération d'une carte de points chauds des densités de contact

Dans cette section, nous présentons une méthodologie nous permettant dans un premier temps de générer une carte de points chauds représentant les variations spatiales des densités de contacts entre les bus. Dans un second temps, nous déterminons les emplacements des zones avec des durées et fréquences de contacts entre bus plus importantes qu'ailleurs sur la carte.



(a) Carte de points chauds de tous les contacts entre bus, quelle que soit leur durée.



(b) Carte de points chauds des contacts de durée supérieure à 200 secondes. Les points chauds représentés par des cercles rouges contiennent des cellules dont la densité de contacts est dans le top 25%.

Figure B.18. Carte de points chauds des contacts entre les bus de Dublin en Irlande. La figure montre le nombre de contacts par cellule de taille 100 m $\times$ 100 m.

Carte de points chauds de contacts. La première étape consiste à calculer les variations spatiales des densités de contacts entre bus. Pour ce faire, la carte est divisée en une grille de cellules carrées de taille 100 m $\times$ 100 m. Pour chaque cellule, nous associons la densité de contacts qui occurrent dans la zone géographique de la cellule. Nous représentons la carte de points chauds montrant la densité de contacts sur la Figure B.18a. Cependant, nous nous intéressons à la migration de machines virtuelles entre bus. Or une machine virtuelle peut avoir une taille relativement importante, nécessitant un transfert de longue durée entre bus. Ainsi, nous représentons sur la Figure B.18b la carte de points chauds des densités de contacts entre bus d'une durée supérieure ou égale à 200 secondes, pouvant alors transférer une machine virtuelle de taille 200 Mégaoctets avec un débit de 1 Mégaoctet par seconde. Nous utilisons une estimation par noyau (*kernel density estimation* en anglais) pour représenter les cellules ayant une haute signification statistique de leur densité de contacts.

Les cartes de points chauds pour la ville de Dublin représentées sur les Figures B.18a et B.18b montrent les densités de contacts entre bus en utilisant différentes teintes de rouge, les teintes plus foncées indiquant une densité plus élevée de contacts. Notons que les contacts sont surtout concentrés dans le centre ville, mais aussi en périphérie, où les bus terminent généralement leur service.

Zones de contact. La deuxième étape consiste à déduire les zones de contact à partir de la carte de points chauds des densités de contacts. Dans un premier temps, nous ordonnons les cellules de la carte de points chauds selon leur densité de contacts. Afin d'exclure les cellules où le nombre de contacts est négligeable, nous sélectionnons les cellules présentant au moins 25% des meilleures densités de contacts. Les cellules contiguës sont ensuite agrégées en ce que l'on appelle des "zones de contacts". Celles-ci sont représentées par des cercles recouvrant l'ensemble des cellules sélectionnées sur la carte de la Figure B.18b, pour des cellules mesurant la densité de contacts de durée supérieure ou égale à 200 secondes. Le diamètre des zones de contacts correspond à celui couvrant le plus grand ensemble de cellules contiguës.

Caractérisation pour les migrations de machines virtuelles. Nous caractérisons ensuite les zones de contact de la Figure B.18b par rapport à la migration de machines virtuelles. Pour chaque zone de contacts, nous calculons les deux variables suivantes : (*i*) le nombre total n de contacts d'une durée supérieure ou égale à 200 secondes et (*ii*) la durée moyenne x entre deux contacts successifs. A partir de ces deux variables, nous calculons un score pour chacun des points chauds de la manière suivante : $n/\max(1,x)$. Ce score mesure la capacité d'une zone a pouvoir traiter fréquemment des transferts nécessitant au plus 200 secondes. Enfin, nous utilisons la méthode d'optimisation de Jenks pour classer les zones de contacts en trois classes de valeurs représentatives.

Dans la Figure B.18b, nous utilisons différentes teintes de couleurs pour représenter les zones de contacts selon leur classe de valeurs. Pour évaluer notre méthodologie, nous sélectionnons deux zones de contacts, chacune appartenant à deux classes différentes. La "zone de contact 1" a le meilleur score, ce qui indique des contacts de longue durée et fréquents (les trois autres zones de contacts ont un score similaire). La "zone de contact 2" est sélectionnée arbitrairement parmi les autres zones ayant un score similaire.

B.3.2.2 Expérimentation avec des traces réalistes

Dans cette section, nous présentons les résultats des simulations que nous avons faites pour montrer la faisabilité des migrations de machines virtuelles via les communications bus-à-bus. Celles-ci sont présentées pour les bus de la ville de Dublin en Irlande. Plus précisément, nous intéresserons à la capacité des contacts entre bus pour transférer des machines virtuelles de taille variables.

Mise en place de l'expérimentation. Nous utilisons un jeu de données publiques des traces de bus de la ville de Dublin, fournies par DublinBus, faisant partie du Dublin City Council. Les traces de mobilité du mois de janvier 2013 donnent les coordonnées GPS estampillées temporellement successives de chaque bus en service opéré par DublinBus [Dub13]. Nous analysons les mouvements de 823 bus pour le service opéré le 29 janvier 2013 entre 10h et 13h (à heure creuse). Nous supposons que deux bus sont en contact s'il sont chacun dans la portée de communication de l'autre. Pour reproduire les mouvements des bus DublinBus et inférer leurs contacts, nous utilisons le simulateur à évènements discrets ONE pour simuler des interfaces IEEE 802.11 sur les bus [KOK09]. Pour s'affranchir des propriétés de la couche physique telles que le taux d'erreurs binaires, nous considérons une portée de communication de rayon 150 m, volontairement pessimiste avec un débit de 1 Mégaoctet par seconde.

Nous simulons les migrations de machines virtuelles pour différentes tailles exécutées par des bus entrant dans les zones de contacts entourées sur la carte de points chauds de Dublin représentée sur la Figure B.18b. Nous considérons un algorithme de migration naïf qui ne fait aucune supposition ni prédiction sur la future durée des contacts : lorsqu'un bus entre dans une zone de contacts, celui-ci initie une migration de machine virtuelle avec le premier bus qu'il rencontre.

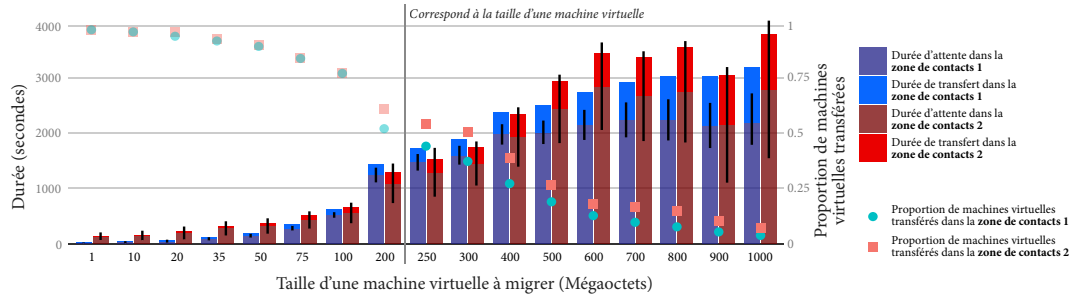


Figure B.19. Proportion des migrations de machine virtuelle en succès et durée moyenne pour transférer les machines virtuelles en fonction de leur taille. Les barres noires représentent les intervalles de confiance à 95% du temps d'attente avant de pouvoir transférer avec succès les machines virtuelles.

La migration est annulée si la durée du contact n'est pas suffisante pour transférer l'intégralité de la machine virtuelle.

Les résultats que nous présentons reflètent les moyennes pour 100 tentatives de migration de machines virtuelles au niveau de chacune des deux zones de contacts. Les différentes tailles que l'on considère représentent les tailles types de machines virtuelles allant de quelques Mégaoctets (*e.g.*, TinyOS [LMP+05]) à plusieurs centaines de Mégaoctets [CFH+05]. Pour chacune des tailles de machine virtuelle, nous mesurons la durée moyenne nécessaire pour qu'un bus qui entre une zone de contact trouve un contact qui durera suffisamment longtemps pour transférer la machine virtuelle. Le graphe de la Figure B.19 montre la durée moyenne à attendre avant de transférer des machines virtuelles de tailles différentes, ainsi que la proportion de machines virtuelles qui ont pu être transférées parmi toutes les migrations initiées.

Tout d'abord, nous remarquons que le ratio de machines virtuelles migrées avec succès décroit avec leur taille, et ce pour chacune des deux zones de contact. Ceci est dû au faible nombre de contacts de longue durée entre les bus, nombre qui décroit avec la durée des contacts. En effet, les bus échouent plusieurs fois avant de trouver un contact qui leur permettra de transférer la machine virtuelle. Cette observation est confirmée par la durée à attendre un "bon" contact pour des machines virtuelles de grandes tailles. En général, cela prend moins de temps à trouver un "bon" contact au niveau de la zone de contacts 1 qu'au niveau de la zone de contacts 2, sauf pour les machines virtuelles dont la taille est comprise entre 200 et 400 Mégaoctets.

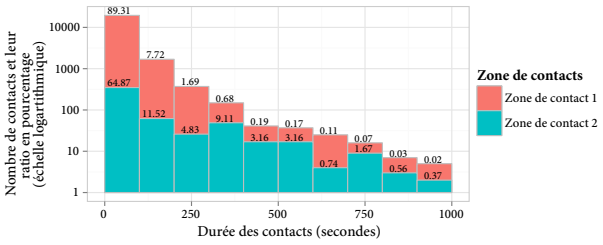


Figure B.20. Nombre de contacts (avec une échelle logarithmique) pour différents intervalles de durées et leur ratios respectifs (en pourcentage) par rapport au nombre total de contacts dans les zones de contacts 1 et 2.

Pour expliquer cette tendance, nous représentons dans la Figure B.20 le nombre total de contacts mesuré pour différentes durées de contacts ayant lieu dans les zones de contacts 1 et 2. Nous pouvons constater que la zone de contacts 2 a un ratio plus élevé de contacts de durées comprises entre 200 et 400 secondes. Malgré leur plus grand nombre dans la zone de contacts 1, la plupart

d'entre eux ne peuvent pas transférer des tailles de machines virtuelles comprises entre 200 et 400 Mégaoctets. Les bus passent alors plus de temps dans la zone de contacts 1 à essayer de transférer de grandes quantités de données qui vont le plus souvent échouer avant de trouver un "bon" contact. Cela explique aussi pourquoi le ratio de migration de machines virtuelles est meilleur dans la zone de contacts 2 lorsque le machines virtuelles ont des tailles comprises entre 200 et 400 Mégaoctets.

B.4 | Conclusions

Le travail présenté dans cette thèse se repose sur la mobilité d'entités de tous les jours, et plus particulièrement, celles des voitures particulières et des bus, afin de s'affranchir des limitations et de la dépendance vis-à-vis des réseaux de données traditionnels. Tout au long de cette thèse, nous avons exploité cette mobilité dans le contexte de différentes applications, dont le délestage massif de données et des services véhiculaires de type "cloud".

Délestage de données massif. Nous avons équipé des véhicules particuliers avec des capacités de stockage connectant des endroits spécifiques que nous avons appelé "point de délestage". Ils permettent de capturer d'avantage de trajets que seulement ceux qui effectuent le trajet complet entre la source et la destination d'un transfert de données. Ces points de délestage aident à maximiser les ressources véhiculaires, c'est-à-dire, l'utilisation de l'espace de stockage combiné des véhicules connectant les points de délestage. Pour ce faire, ils jouent le rôle de relais où les trajets des véhicules s'y arrêtant sont composées en un seul chemin logique suivi par le cargo de données qu'ils transportent. Dans la suite, nous détaillons comment nous avons utilisé ces points de délestage pour contrôler et configurer les mouvements des véhicules afin de satisfaire les besoins des transferts de données.

Contrôle et configuration de l'infrastructure de délestage. Les points de délestage sont également équipés de capacités de stockage où les cargos de données sont temporairement stockés et passés de manière asynchrone d'un véhicule à l'autre. Ils sont placés là où les véhicules s'arrêtent suffisamment longtemps pour transférer de grandes quantités de données, par exemple dans les parkings situés le long des routes ou ceux situés dans des villes ou des supermarchés, ou encore des stations-services d'essence ou de chargement de batteries électriques. Nous avons proposé une architecture de type SDN (*Software-Defined Networking*) comprenant un contrôleur centralisé qui a une vue globale de l'infrastructure de délestage. Le contrôleur configure les ressources véhiculaires de cette infrastructure à l'aide du réseau de délestage, une représentation logique en quantité réseau des flots de véhicules entre les points de délestage. Plus précisément, ce réseau comporte des liens caractérisés par une capacité, un temps de trajet moyen et un taux d'erreur. Le contrôleur utilise ce réseau de délestage pour résoudre le problème d'allocation de flots de véhicules, qui consiste à sélectionner les chemins logiques du réseau de délestage et à assigner (allouer) des quantités de données sur chacun de ces chemins de façon à satisfaire les besoins spécifiques aux transferts de données. La résolution du problème d'allocation maximise l'utilisation des ressources tout en garantissant une équité entre les allocations des

transferts. Son résultat est traduit en règles pour configurer les points de délestage. Ces règles permettent de sélectionner les cargos de données à charger sur les véhicules à l'arrêt. Nos résultats sur le réseau routier français avec des comptages de trafic réels ont permis de montrer que cette architecture centralisée permet d'effectuer une allocation efficiente et équitable des ressources pour des transferts de données en compétition entre des grandes villes de France.

Système de stockage et partage véhiculaire. Nous avons exploité le stockage des points de délestage pour les transformer en dépôts où des utilisateurs mobiles peuvent stocker et récupérer des fichiers. Le système repose sur un ensemble de dépôts et les mouvements de utilisateurs entre ceux-ci pour distribuer les fichier dans l'ensemble du système. Nous proposons un algorithme de placement qui permet de déterminer les emplacements optimaux des dépôts de telle sorte que chaque dépôt puisse satisfaire les requêtes des utilisateurs pour stocker ou récupérer des fichiers. Le placement des dépôts prend également en compte les mouvements des utilisateurs afin que les fichiers soient synchronisés entre les dépôts du système sans avoir à recourir à un réseau de données. Nous avons montré que la distribution des fichiers en utilisant le mouvement des utilisateurs mobiles a permis d'augmenter le taux de succès des requêtes.

Réseaux véhiculaires virtuels. Nous avons dématérialisé les points de délestage dans le contexte de réseaux véhiculaires où les ressources hébergées par les véhicules (*i.e.*, calcul, stockage et communications) sont virtualisées avec des machines virtuelles. Nous avons étudié les migrations de machines virtuelles par des communications véhicule-à-véhicules au lieu de réseau de données de type cellulaires. Dans ce contexte, les points de délestage correspondent aux zones où les véhicules sont fréquemment en contact pour des durées suffisamment longues de manière à pouvoir transférer de grandes quantités de données. Nous avons montré que restreindre les migrations de machines virtuelles à ces zones a permis d'améliorer le taux de succès des migrations de machines virtuelles.

Pour résumer et conclure, les points importants de cette thèse sont les suivants :

- Les véhicules en circulation constituent une ressource avec un potentiel encore inexploité pour transporter des données. Nous avons illustré cette affirmation en simulant des véhicules transportant des données sur les routes françaises et nous avons montré que le réseau routier a le potentiel de transporter plusieurs Pétaoctets de données par jour.
- Une infrastructure dédiée augmente la capacité du système en composant les trajets des véhicules et en configurant le chemin suivi par les données.
- Un contrôle centralisé de l'infrastructure bénéficiant d'une vue globale des flots de véhicules permet une gestion et une allocation efficiente des ressources aux transferts de données.
- Une représentation logique des flots de véhicules en quantités réseau permet de pallier à la complexité du réseau routier et rend l'allocation de transfert de données possible.
- La mobilité des entités du quotidien a le potentiel de fournir des services avec une dépendance limitée vis-à-vis des réseaux de données traditionnels. Nous avons illustré cette affirmation en étendant notre système de délestage dans le contexte de services véhiculaires de type "cloud".

Acronyms

AADT Annual Average Daily Traffic 34, 36, 43–45, 66, 72, 97, 99, 100
AUV Autonomous Underwater Vehicle 14
AWS Amazon Web Services 1, 16
BPR Bureau of Public Roads 35, 45
CAPEX Capital EXpenditure Costs 38
CPLEX IBM ILOG CPLEX Optimization Studio 45
DTN Delay-Tolerant Network 9
ENTD Enquête Nationale Transports et Déplacements 43, 44, 99
EV Electric Vehicle 29, 32
FHWA Ferderal Highway Administration 72
FIMF Ferry-Initiated Message Ferry 22, 23
HPMS Highway Performance Monitoring System 72
IP Internet Protocol 1
ISP Internet Service Provider 15, 37, 94
LP Linear Programing 30, 38, 57, 62
LSA Link State Announcement 25
MANET Mobile Ad hoc Network 9
MAV Micro Aerial Vehicle 23
MCF Multi-Commodity Flow 38, 65
MULE Mobile Ubiquitous LAN Extension 17, 18
NHTS National Household Travel Survey 99
NIMF Node-Initiated Message Ferry 22, 23
O-D Origin-Destination 67, 73, 100
RAID Redundant Array of Inexpensive Disks 36, 59–62
SDN Software-Defined Networking 53, 54, 56, 73, 91
SR-ARQ Selective Repeat Automatic Repeat reQuest 59–61
SUMO Simulation of Urban MObility 55, 67
TSP Travel Salesman Problem 13, 14, 22
V2V Vehicle-to-Vehicle 84, 85, 90
VANET Vehicular Ad hoc Network 9, 20
WSN Wireless Sensor Network 9

Alphabetical index

Bibliography

[ABB+03] William Allcock, Joe Bester, John Bresnahan, Ann Chervenak, Lee Liming, and Steve Tuecke. Gridftp: Protocol extensions to ftp for the grid. *Global Grid ForumGFD-RP*, 2003.

[ADGW13] Ittai Abraham, Daniel Delling, Andrew V Goldberg, and Renato F Werneck. Alternative routes in road networks. *ACM Journal of Experimental Algorithmics (JEA)*, 18:1.3:1.1–1.3:1.17, 2013.

[ADTB15] Mehdi Anteur, Vincent Deslandes, Nathalie Thomas, and Andre-Luc Beylot. Ultra narrow band technique for low power wide area communications. In *IEEE GLOBECOM*, San Diego, CA, USA, December 2015.

[AGH+13] Mahdi Asadpour, Domenico Giustiniano, Karin Anna Hummel, Simon Heimlicher, and Simon Egli. Now or later?: delaying data transfer in time-critical aerial communication. In *ACM CoNEXT*, Santa Barbara, CA, USA, December 2013.

[AHGD16] Mahdi Asadpour, Karin A Hummel, Domenico Giustiniano, and Stefan Draskovic. Route or carry: Motion-driven packet forwarding in micro aerial vehicle networks. *IEEE Transactions on Mobile Computing*, 2016.

[APM05] Ian F Akyildiz, Dario Pompili, and Tommaso Melodia. Underwater acoustic sensor networks: research challenges. *Elsevier Ad hoc networks*, 3(3):257–279, 2005.

[ASLF14] Elian Aubry, Thomas Silverston, Abdelkader Lahmadi, and Olivier Festor. Crowdout: a mobile crowdsourcing service for road safety in digital cities. In *IEEE PERCOM Workshops*, Budapest, Hungary, March 2014.

[AVdBG+14] Mahdi Asadpour, Bertold Van den Bergh, Domenico Giustiniano, Karin Anna Hummel, Sofie Pollin, and Bernhard Plattner. Micro aerial vehicle networks: an experimental analysis of challenges and opportunities. *IEEE Communications Magazine*, 52(7):141–149, 2014.

[BBB04] Jenna Burrell, Tim Brooke, and Richard Beckwith. Vineyard computing: Sensor networks in agricultural production. *IEEE Pervasive Computing*, 3(1):38–45, 2004.

[BBEK11] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo-simulation of urban mobility – an overview. In *SIMUL 2011, The Third International Conference on Advances in System Simulation*, 2011.

[BBL05] Brendan Burns, Oliver Brock, and Brian Neil Levine. MV routing and capacity

building in disruption tolerant networks. In *IEEE INFOCOM*, Miami, FL, USA, March 2005.

[BBL06] Brendan Burns, Oliver Brock, and Brian Neil Levine. Autonomous enhancement of disruption tolerant networks. In *IEEE ICRA*, Orlando, FL, USA, May 2006.

[BBL08] Brendan Burns, Oliver Brock, and Brian Neil Levine. MORA routing and capacity building in disruption-tolerant networks. *Elsevier Ad hoc networks*, 6(4):600–620, 2008.

[BC04] Michel Bierlaire and Frank Crittin. An efficient algorithm for real-time estimation and prediction of dynamic od tables. *INFORMS Operations Research*, 52(1):116–127, 2004.

[BCL07] Nilanjan Banerjee, Mark D Corner, and Brian Neil Levine. An energy-efficient architecture for DTN throwboxes. In *IEEE INFOCOM*, Anchorage, AK, USA, May 2007.

[BGJL06] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. MaxProp: Routing for vehicle-based disruption-tolerant networks. In *IEEE INFOCOM*, Barcelona, Spain, April 2006.

[BLV10] Aruna Balasubramanian, Brian Neil Levine, and Arun Venkataramani. Replication routing in DTNs: a resource allocation approach. *IEEE/ACM Transactions on Networking*, 18(2):596–609, 2010.

[BMZA12] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *ACM MCC*, Helsinki, Finland, August 2012.

[Bre67] Lev M Bregman. Proof of the convergence of sheleikhovskii's method for a problem with transportation constraints. *Elsevier USSR Computational Mathematics and Mathematical Physics*, 7(1):191–204, 1967.

[BTAZ06] Muhammad Mukarram Bin Tariq, Mostafa Ammar, and Ellen Zegura. Message ferry route design for sparse ad hoc networks with mobile nodes. In *ACM MobiHoc*, Florence, Italy, May 2006.

[CCD+04] Scott Carter, Elizabeth Churchill, Laurent Denoue, Jonathan Helfman, and Les Nelson. Digital graffiti: public annotation of multimedia content. In *ACM CHI*, Vienna, Austria, April 2004.

[CFH+05] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *USENIX NSDI*, Boston, MA, USA, May 2005.

[CG10] Brian Cho and Indranil Gupta. New algorithms for planning bulk transfer via internet and shipping networks. In *IEEE ICDCS*, Genoa, Italy, June 2010.

[CG11] Brian Cho and Indranil Gupta. Budget-constrained bulk data transfer via internet and shipping networks. In *ACM ICAC*, Karlsruhe, Germany, June 2011.

[Cha06] Sudarshan S Chawathe. Inter-vehicle data dissemination in sparse equipped traffic. In *IEEE ITSC*, Toronto, ON, Canada, September 2006.

[CIM93] Ennio Cascetta, Domenico Inaudi, and Gerald Marquis. Dynamic estimators of origin-destination matrices using traffic counts. *INFORMS Transportation science*, 27(4):363–373, 1993.

[Cis16] Cisco Visual Networking Index (VNI). Forecast and methodology, 2015-2020, June 2016.

[CLG+94] Peter M Chen, Edward K Lee, Garth A Gibson, Randy H Katz, and David A Patterson. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, 1994.

[CLGH10] Pei-Chun Cheng, Kevin C Lee, Mario Gerla, and Jérôme Härri. GeoDTN+ Nav: Geographic DTN routing with navigator prediction for urban vehicular environments. *Springer Mobile Networks and Applications*, 15(1):61–82, 2010.

[CN96] Ennio Cascetta and Agostino Nuzzolo. A modified logit route choice model overcoming path overlapping problems: specification and some calibration results for interurban networks. In *Internaional symposium on transportation and traffic theory*, Lyon, France, July 1996.

[CV74] Richard Church and CHARLES R VELLE. The maximal covering location problem. *Wiley Papers in regional science*, 32(1):101–118, 1974.

[DCVR06] Matthew Dunbabin, Peter Corke, Iuliu Vasilescu, and Daniela Rus. Data muling over underwater wireless sensor networks using an autonomous underwater vehicle. In *IEEE ICRA 2006*, Orlando, FL, USA, May 2006.

[DDB08] Michael J Demmer, Bowei Du, and Eric A Brewer. TierStore: A distributed filesystem for challenged networks in developing regions. In *USENIX FAST*, San Jose, CA, USA, February 2008.

[dDOW11] Juan de Dios Ortúzar and Luis Willumsen. *Modelling transport*. Wiley Chichester, 2011.

[DF07] Michael Demmer and Kevin Fall. DTLSR: delay tolerant routing for developing regions. In *ACM NSDR*, Kyoto, Japan, August 2007.

[Dia71] Robert B Dial. A probabilistic multipath traffic assignment model which obviates path enumeration. *Pergamon Transportation research*, 5(2):83–111, 1971.

[Dij59] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Springer Numerische mathematik*, 1(1):269–271, 1959.

[DS77] Carlos F Daganzo and Yosef Sheffi. On stochastic models of traffic assignment. *INFORMS Transportation science*, 11(3):253–274, 1977.

[DSSW09] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Engineering route planning algorithms. In *Springer Algorithmics of large and complex networks*. 2009.

[Dub13] Dublinked. Dublin bus gps sample data from dublin city council (insight project). http://dublinked.com/datastore/datasets/dataset-304.php, 2013. Accessed: September 22, 2016.

[EML+09] Shane B Eisenman, Emiliano Miluzzo, Nicholas D Lane, Ronald A Peterson, Gahng-Seop Ahn, and Andrew T Campbell. Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks*, 6(1):6:1–6:39, 2009.

[Epp98] David Eppstein. Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673, 1998.

[FJF15] Lester Randolph Ford Jr and Delbert Ray Fulkerson. *Flows in networks*. Princeton university press, 2015.

[FS07] Lisa Fleischer and Martin Skutella. Quickest flows over time. *SIAM Journal on Computing*, 36(6):1600–1630, 2007.

[GBMY97] David J Goodman, Joan Borras, Narayan B Mandayam, and Roy D Yates. Infostations: A new system model for data and messaging services. In *IEEE VTC*, Phoenix, AZ, USA, May 1997.

[GDF+11] Shimin Guo, Mohammad Derakhshani, Mohammad Hossein Falaki, Usman Ismail, Rowe Luk, Earl A Oliver, S Ur Rahman, Aaditeshwar Seth, Matei A Zaharia, and Srinivasan Keshav. Design and implementation of the KioskNet system. *Elsevier Computer Networks*, 55(1):264–281, 2011.

[GFO+07] Shimin Guo, Mohammad Hossein Falaki, Earl A Oliver, S Ur Rahman, Aaditeshwar Seth, Matei A Zaharia, and Srinivasan Keshav. Very low-cost internet access using KioskNet. *ACM SIGCOMM Computer Communication Review*, 37(5):95–100, 2007.

[GKS10] Robert Geisberger, Moritz Kobitzsch, and Peter Sanders. Route planning with flexible objective functions. In *SIAM ALENEX*, Austin, Texas, January 2010.

[GO92] Arthur Getis and J Keith Ord. The analysis of spatial association by use of distance statistics. *Wiley Geographical analysis*, 24(3):189–206, 1992.

[GR12] John Gantz and David Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the Future*, 2012.

[GT01] Matthias Grossglauser and David Tse. Mobility increases the capacity of adhoc wireless networks. In *IEEE INFOCOM*, Anchorage, AK, USA, April 2001.

[Gui13] Traffic Monitoring Guide. Us department of transportation. *Federal Highway Administration*, 2013.

[GV06] Matthias Grossglauser and Martin Vetterli. Locating mobile nodes with ease: learning efficient routes from encounter histories alone. *IEEE/ACM Transactions on Networking*, 14(3):457–469, 2006.

[HBZ+06] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. Cartel: a distributed mobile sensor computing system. In *ACM SenSys*, Boulder, CO, USA, October 2006.

[HCY11] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: Social-based forwarding in delay-tolerant networks. *IEEE Transactions on Mobile Computing*, 10(11):1576–1589, 2011.

[Hec16] Jeff Hecht. The bandwidth bottleneck that is throttling the Internet. *Nature*, 536(7615):139, August 2016.

[HKBA07] David Hadaller, Srinivasan Keshav, Tim Brecht, and Shubham Agarwal. Vehicular opportunistic communication under the microscope. In *ACM MobiSys*, San Juan, Puerto Rico, June 2007.

[HNR68] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[Hod90] M John Hodgson. A flow-capturing location-allocation model. *Wiley Geographical Analysis*, 22(3):270–279, 1990.

[INC09] Mouhamad Ibrahim, Philippe Nain, and Iacopo Carreras. Analysis of relay protocols for throwbox-equipped DTNs. In *IEEE WiOPT*, Seoul, Korea, June 2009.

[Jen63] George F Jenks. Generalization in statistical mapping. *Taylor & Francis Annals of the Association of American Geographers*, 53(1):15–26, 1963.

[JLW+16] Xin Jin, Yiran Li, Da Wei, Siming Li, Jie Gao, Lei Xu, Guangzhi Li, Wei Xu, and Jennifer Rexford. Optimizing bulk transfers with software-defined optical wan. In *ACM SIGCOMM*, Florianópolis, Brazil, August 2016.

[JOW+02] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. *ACM Sigplan Notices*, 37(10):96–107, 2002.

[JSRGD09] Moez Jerbi, Sidi-Mohammed Senouci, Tinku Rasheed, and Yacine Ghamri-Doudane. Towards efficient geographic routing in urban vehicular networks. *IEEE Transactions on Vehicular Technology*, 58(9):5048–5059, 2009.

[Kar08] George Karakostas. Faster approximation schemes for fractional multicommodity flow problems. *ACM Transactions on Algorithms*, 4(1):13, 2008.

[KH06] John Krumm and Eric Horvitz. Predestination: Inferring destinations from partial trajectories. In *Springer UbiComp*, pages 243–260. 2006.

[KO09] Ari Keränen and Jörg Ott. DTN over aerial carriers. In *ACM CHANTS*, Beijing, China, September 2009.

[KOK09] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE simulator for DTN protocol evaluation. In *ACM Simutools*, Rome, Italy, March 2009.

[KWSB04] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello. Extracting places from traces of locations. In *ACM WMASH*, Philadelphia, PA, USA, October 2004.

[LC04] Shu Lin and Daniel J Costello. *Error control coding*. Prentice-hall Englewood Cliffs, 2004.

[LDP+07] Sven Lahde, Michael Doering, Wolf-Bastian Pöttner, Gerrit Lammert, and Lars Wolf. A practical analysis of communication characteristics for mobile and distributed pollution measurements on the road. *Wiley Wireless Communications and Mobile Computing*, 7(10):1209–1218, 2007.

[LDS03] Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE mobile computing and communications review*, 7(3):19–20, 2003.

[Lee61] Chin Yang Lee. An algorithm for path connections and its applications. *IRE transactions on electronic computers*, (3):346–365, 1961.

[Lim14] Infiniti Research Limited. Global charging equipment for ev market 2014-2018, 2014.

[LJ10] T Le Jeannic. La mobilité des français, panorama issu de l'enquête nationale transports et déplacements 2008. *Paris: ministère de l'Écologie, du Développement durable, des Transports et du Logement*, 2010.

[LLHG08] Kevin C Lee, Michael Le, Jerome Harri, and Mario Gerla. LOUVRE: Landmark overlays for urban vehicular routing environments. In *IEEE VTC Fall*, Calgary, BC, Canada, September 2008.

[LM07] Ilias Leontiadis and Cecilia Mascolo. Geopps: Geographical opportunistic routing for vehicular networks. In *IEEE WoWMoM*, Espoo, Finland, June 2007.

[LMP+05] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, et al. Tinyos: An operating system for sensor networks. In *Springer Ambient intelligence*. 2005.

[LPFK07] Lin Liao, Donald J Patterson, Dieter Fox, and Henry Kautz. Learning and inferring transportation routines. *Elsevier Artificial Intelligence*, 171(5):311–331, 2007.

[LSRS09] Nikolaos Laoutaris, Georgios Smaragdakis, Pablo Rodriguez, and Ravi Sundaram. Delay tolerant bulk data transfers on the internet. In *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, ACM SIGMETRICS, Seattle, WA, USA, June 2009.

[LSS+13] Nikolaos Laoutaris, Georgios Smaragdakis, Rade Stanojevic, Pablo Rodriguez,

and Ravi Sundaram. Delay-tolerant bulk data transfers on the internet. *IEEE/ACM Transactions on Networking*, 21(6):1852–1865, January 2013.

[LSYR11] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez. Inter-datacenter bulk transfers with NetStitcher. In *ACM SIGCOMM*, Toronto, ON, Canada, August 2011.

[MAB+08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[MAZ11] Ahmed Mansy, Mostafa Ammar, and Ellen Zegura. Deficit round-robin based message ferry routing. In *IEEE GLOBECOM*, Houston, TX, USA, December 2011.

[MED] MEDDE. Mobilité – déplacement. `http://tinyurl.com/otfbewv`.

[MGH+07] P. McDonald, D. Geraghty, I. Humphreys, S. Farrell, and V. Cahill. Sensor network with delay tolerance (sendt). In *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, 2007.

[MS77] F MacWilliams and NJ Sloane. *The theory of rrror-correcting codes*. Elsevier, 1977.

[MZH83] Nimrod Megiddo, Eitan Zemel, and S Louis Hakimi. The maximum coverage location problem. *SIAM Journal on Algebraic Discrete Methods*, 4(2):253–261, 1983.

[Nat00] National Research Council (U.S.). Transportation Research Board. *Highway Capacity Manual*. 2000.

[NF13] Diala Naboulsi and Marco Fiore. On the instantaneous topology of a large-scale urban vehicular network: the cologne case. In *ACM MobiHoc*, Bangalore, India, July 2013.

[NK09] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *ACM SIGSPATIAL*, Seattle, WA, USA, November 2009.

[OHL+11] Jörg Ott, Esa Hyytiä, Pasi Lassila, Tobias Vaegs, and Jussi Kangasharju. Floating content: Information sharing in urban areas. In *IEEE PerCom*, Seattle, WA, USA, March 2011.

[OP07] Jörg Ott and Mikko Juhani Pitkänen. DTN-based content storage and retrieval. In *IEEE WoWMoM*, Espoo, Finland, June 2007.

[Pat03] Dave Patterson. A conversation with jim gray. *ACM Queue*, 1(4):53–56, July 2003.

[PFH04] Alex Sandy Pentland, Richard Fletcher, and Amir Hasson. Daknet: Rethinking connectivity in developing nations. *IEEE Computer*, 37(1):78–83, January 2004.

[PKL07] Jim Partan, Jim Kurose, and Brian Neil Levine. A survey of practical issues in underwater networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 11(4):23–33, 2007.

[PNKF03] Michal Pióro, Pål Nilsson, Eligijus Kubilinskas, and Gábor Fodor. On efficient max-min fair routing algorithms. In *IEEE ISCC*, Kemer - Antalya, Turkey, June 2003.

[PO07] Mikko Juhani Pitkänen and Jörg Ott. Redundancy and distributed caching in mobile DTNs. In *ACM/IEEE MobiArch*, Kyoto, Japan, August 2007.

[Rab04] A Rabagliati. Wizzy digital courier–how it works, 2004.

[RBADdA+09] Marcelo Gonçalves Rubinstein, Fehmi Ben Abdesslem, M Dias de Amorim, Sávio Rodrigues Cavalcanti, Rafael dos Santos Alves, Luís Henrique Maciel Kosmalski Costa, Otto Carlos Muniz Bandeira Duarte, and Miguel Elias Mitre Campista. Measuring the capacity of in-car to in-car vehicular networks. *IEEE Communications Magazine*, 47(11):128–136, 2009.

[Sav85] Martin WP Savelsbergh. Local search in routing problems with time windows. *Springer Annals of Operations research*, 4(1):285–305, 1985.

[SBZS06] Reid Simmons, Brett Browning, Yilu Zhang, and Varsha Sadekar. Learning to predict driver route and destination intent. In *IEEE ITSC*, Toronto, ON, Canada, September 2006.

[SCHD06] James Scott, Jon Crowcroft, Pan Hui, and Christophe Diot. Haggle: A networking architecture designed around mobile users. In *IEEE/IFIP WONS*, Les Ménuires, France, January 2006.

[SDPG06] Natasa Sarafijanovic-Djukic, Michal Piorkowski, and Matthias Grossglauser. Island hopping: Efficient mobility-assisted forwarding in partitioned networks. In *IEEE SECON*, Reston, VA, USA, September 2006.

[SH03] Tara Small and Zygmunt J Haas. The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In *ACM MobiHoc*, Annapolis, MD, USA, June 2003.

[She85] Y. Sheffi. *Urban transportation networks: equilibrium analysis with mathematical programming methods*. Prentice Hall, 1985.

[Sil86] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[SKZ+06] Aaditeshwar Seth, Darcy Kroeker, Matei Zaharia, Shimin Guo, and Srinivasan Keshav. Low-cost communication for rural internet kiosks using mechanical backhaul. In *ACM MobiCom*, Los Angeles, CA, USA, September 2006.

[SPR05] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *ACM WDTN*, Philadelphia, PA, USA, August 2005.

[SPR07] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S Raghavendra. Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *IEEE PerCom Workshops*, White Plains, NY, USA, March 2007.

[SRF14] Vasco NGJ Soares, Joel JPC Rodrigues, and Farid Farahmand. Geospray: A geographic routing protocol for vehicular delay-tolerant networks. *Elsevier Information Fusion*, 15:102–113, 2014.

[SRJB03] Rahul C Shah, Sumit Roy, Sushant Jain, and Waylon Brunette. Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks. *Elsevier Ad Hoc Networks*, 1(2):215–233, September 2003.

[SV96] Madhavapeddi Shreedhar and George Varghese. Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, 1996.

[TILT09] Onur Tekdas, Volkan Isler, Jong Hyun Lim, and Andreas Terzis. Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications*, 16(1):22, 2009.

[TJWY14] Bo Tan, Jithin Jose, Xinzhou Wu, and Lei Ying. A vehicular backbone network (VBN) with joint transportation-wireless capacity utilization. In *IEEE WiOpt*, Hammamet, Tunisia, May 2014.

[TLLB04] Yuldi Tirta, Zhiyuan Li, Yung-Hsiang Lu, and Saurabh Bagchi. Efficient collection of sensor data in remote fields using mobile collectors. In *IEEE ICCCN 2004*, Chicago, IL, USA, October 2004.

[Uni64] United States. Bureau of Public Roads. *Traffic assignment manual for application with a large, high speed computer*. 1964.

[UTCFBO14] Sandesh Uppoor, Oscar Trullols-Cruces, Marco Fiore, and Jose M Barcelo-Ordinas. Generation and analysis of a large-scale urban vehicular mobility dataset. *IEEE Transactions on Mobile Computing*, 13(5):1061–1075, 2014.

[VB+00] Amin Vahdat, David Becker, et al. Epidemic routing for partially connected ad hoc networks. Technical report, Technical Report CS-200006, Duke University, 2000.

[VZW80] Henk J Van Zuylen and Luis G Willumsen. The most likely trip matrix estimated from traffic counts. *Elsevier Transportation Research Part B: Methodological*, 14(3):281–293, 1980.

[Wai90] D. Waitzman. A standard for the transmission of ip datagrams on avian carriers. RFC 1149, April 1990.

[War52] John Glen Wardrop. Some theoretical aspects of road traffic research. 1952.

[WHYL97] Tommy Wright, Patricia S Hu, Jennifer Young, and An Lu. Variability in traffic monitoring data. 1997.

[WSG+04] Randolph Y Wang, Sumeet Sobti, Nitin Garg, Elisha Ziskind, Junwen Lai, and Arvind Krishnamurthy. Turning the postal system into a generic digital communication mechanism. 34(4):159–166, 2004.

[XLZL09] Guangtao Xue, Zhongwei Li, Hongzi Zhu, and Yunhuai Liu. Traffic-known urban vehicular route prediction based on partial mobility patterns. In *IEEE ICPADS*, Shenzhen, China, December 2009.

[Yen71] Jin Y Yen. Finding the k shortest loopless paths in a network. *INFORMS Management Science*, 17(11):712–716, 1971.

[ZA03] Wenrui Zhao and Mostafa H Ammar. Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks. In *IEEE FTDCS*, San Juan, Puerto Rico, May 2003.

[ZAC10] Mohammad Zarafshan-Araki and Kwan-Wu Chin. Trainnet: A transport system for delivering non real-time data. *Elsevier Computer Communications*, 33(15):1850–1863, 2010.

[ZAZ04] Wenrui Zhao, Mostafa Ammar, and Ellen Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *ACM MobiHoc*, Tokyo, Japan, May 2004.

[ZAZ05] Wemi Zhao, Mostafa Ammar, and Ellen Zegura. Controlling the mobility of multiple data transport ferries in a delay-tolerant network. In *IEEE INFOCOM*, Miami, FL, USA, March 2005.

[ZCA+06] Wenrui Zhao, Yang Chen, Mostafa Amma, Mark Corner, Brian Levine, and Ellen Zegura. Capacity enhancement using throwboxes in DTNs. In *IEEE MASS*, Vancouver, BC, USA, October 2006.

[ZMBD08] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, Chicago, IL, USA, July 2008.

[ZRL+13] Jiazhen Zhou, Sandip Roy, Jiang Li, Rose Hu, and Yi Qian. Minimizing the average delay of messages in pigeon networks. *IEEE Transactions on Communications*, 61(8):3349–3361, 2013.