Centrally-Controlled Mass Data Offloading Using Vehicular Traffic

Benjamin Baron, Prométhée Spathis, Hervé Rivano, Marcelo Dias de Amorim, Yannis Viniotis, Mostafa Ammar

Centrally-Controlled Mass Data Offloading Using Vehicular Traffic

Benjamin Baron, Prométhée Spathis, Hervé Rivano, Marcelo Dias de Amorim,
Yannis Viniotis, and Mostafa Ammar

***Abstract*—With over 300 billion vehicle trips made in the USA and 64 billion in France per year, network operators have the opportunity to utilize the existing road and highway network as an alternative data network to offload large amounts of delay-tolerant traffic. To enable the road network as a large-capacity transmission system, we exploit the existing mobility of vehicles equipped with wireless and storage capacities together with a collection of *offloading spots*. An offloading spot is a data storage equipment located where vehicles usually park. Data is transloaded from a conventional data network to the closest offloading spot and then shipped by vehicles along their line of travel. The subsequent offloading spots act as data relay boxes where vehicles can drop off data for later pick-ups by other vehicles, depending on their direction of travel. The main challenges of this offloading system are how to compute the road path matching the performance requirements of a data transfer and how to configure the sequence of offloading spots involved in the transfer. We propose a scalable and adaptive centralized architecture built on SDN that maximizes the utilization of the flow of vehicles connecting consecutive offloading spots. We simulate the performance of our system using real roads traffic counts for France. Results show that the centralized controlled offloading architecture can achieve an efficient and fair allocation of concurrent data transfers between major cities in France.**

***Index Terms*—Offloading, Software-Defined Networking, Vehicular Data Backhaul.**

I. Introduction

We consider a large-scale data offloading system that takes opportunistic advantage of the mobility of conventional vehicles to transfer massive amounts of delay-tolerant traffic using the road network. By leveraging a large number of daily journeys involving vehicles, content providers or network operators can alleviate the traffic load from conventional data networks such as the Internet. In a previous work [1], [2], we proposed to equip conventional vehicles with removable and exchangeable storage devices as well as wireless interfaces turning the vehicles into data carriers while making their routine journeys.[1] The scale of the road network, together with the number of trips traveled, enables vehicles to transport data in massive amounts and over long distances.

Our offloading system aims at the growth of data traffic which might bring the Internet to a "capacity crunch" in a near future [3], [4]. With the enhancement of access networks and the demand increase in data traffic, CDN providers such as Akamai have reported that the bottleneck is no longer at the origin or the destination of the transfers, but could also be at the core of the network, including peering points between Internet Service Providers and within the providers' networks [5]. Furthermore, as reported by Hecht, geo-distributed services are one of the biggest drivers of demand for bandwidth, as they require transfers of massive amounts of data for synchronization and maintenance [4].

Our system exploits the large number and wide coverage of the trips made by private vehicles to extend the capacity of conventional data networks, while avoiding costly infrastructure upgrades. In particular, we target services that can tolerate transfers lasting several days. Examples of such services include bandwidth-intensive background services such as maintenance activities, data migration, and offline backups [6].

We complement the role of the storage-enabled vehicles with the use of *offloading spots*. Offloading spots are data storage equipments placed along the roads where vehicles usually park long enough as part of their daily routines. Examples of such locations are on-street parking spots, garage parking, gas and electric charging stations, or supermarket parking lots. Offloading spots serve two distinct purposes depending on their relative position in respect to the offloading process. The dual role of an offloading spot is depicted in Figure 1, where massive amount of delay-tolerant background data needs to be transferred between two remote data centers. Part of or all the data originating from the data network is first *transloaded*[2] to the closest edge offloading spot until transferred onto passing vehicles. Subsequent intermediate offloading spots then act as data relay boxes where data is *transshipped*[3] between successive vehicles, allowing the vehicles to drop off their data cargo as part of their route for later pick-up by another vehicle. Once at the destination, the data is transloaded back into the original data network. Since transloading takes place over short distances, it can be carried out by dedicated vehicles in charge of transporting data

Benjamin Baron, Prométhée Spathis, and Marcelo Dias de Amorim are with the LIP6/CNRS Computer Science laboratory, Université Pierre et Marie Curie, Paris, France. Emails: {bbaron,spathis,amorim}@npa.lip6.fr. Hervé Rivano is with Inria, Université de Lyon, INSA-Lyon/CITI, Villeurbanne, France. Email: herve.rivano@inria.fr. Yannis Viniotis is with NC State University. Email: candice@ncsu.edu. Mostafa Ammar is with Georgia Tech. Email: ammar@cc.gatech.edu. This author's work was supported in part by NSF grant NETS 1409589.

A preliminary version of this paper appeared as: B. Baron, P. Spathis, H. Rivano, M. Dias de Amorim, Y. Viniotis, and J. Clarke, "Software-Defined Vehicular Backhaul", *Wireless Days*, Rio de Janeiro, Brazil, Nov. 2014.

[1] A related system for large-scale data transport using removable storage devices is used in the Amazon AWS Snowball system: https://aws.amazon.com/importexport/.

[2] Objects are said to be "transloaded" if they are transferred between two different modes of transportation – from the Internet to moving vehicles in this case.

[3] Objects are said to be "transhipped" if they are moved between similar carriers – from one vehicle to another in this case.

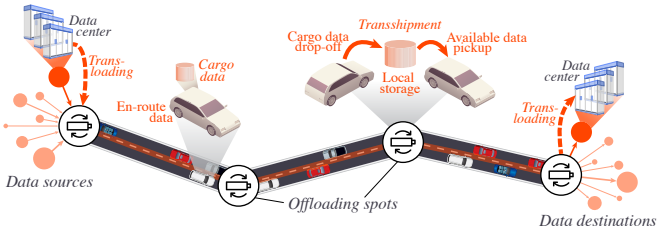

Fig. 1: Overview of our vehicular data offloading system in the context of a data transfer between two remote data centers offloaded from a conventional data network to the road network.

or even high-throughput dedicated lines in case of continuous data transfers.

To enable efficient and reliable transfers over the road network, we propose a centralized architecture for flexible and scalable configuration of the network of offloading spots. We use the software-defined networking (SDN) paradigm, which provides the necessary logistics for efficient and effective vehicular transportation of data [7]. Our SDN-controlled architecture consists of a central controller and a collection of offloading spots. The controller receives demands to offload data transfers onto the road network. An offloading demand indicates the source and destination of the transfer as well as its performance requirements (*e.g.*, in terms of volume and delay). The controller is in charge of mapping a data transfer onto a sequence of offloading spots matching the direction of a vehicle against the destination of the data.

To realize an efficient data offloading, our SDN-controlled architecture addresses the following challenges. Firstly, we must take all vehicular travels into account, including the travels between intermediate offloading spots. Secondly, we need an architecture that can cope with the complexity of the road network topology and the large number of vehicular trips. Thirdly, we need an efficient allocation process of the *road resources* represented by the flows of vehicles traveling between the offloading spots. This allocation should match the performance requirements of the offloaded data transfers, while guaranteeing a fair distribution of the road resources to the data transfers. Finally, we must guarantee reliable data transfers through retransmissions. Updates of allocation decisions are also required for maintaining high utilization in the face of changes in the road traffic.

We develop two novel algorithms implemented at the controller to realize our centralized architecture. The first algorithm copes with the complexity of the road network and the large number of trips by computing a logical map of the offloading infrastructure from the vehicle flows between the offloading spots. The second algorithm is the vehicle flow allocation that performs a fair allocation of the offloading demands on the logical map. We formulate this algorithm as a max-min fairness allocation problem. By solving this problem, the controller determines the optimal network paths that accommodate the requirements of each offloading demand. A network path consists of a sequence of offloading spots and the road segments connecting them together. The controller dictates the behavior of each offloading spot by installing the forwarding states resulting from the allocation procedure. Forwarding states enable offloading spots to assign the data to vehicles traveling the corresponding network paths such that it guarantees a fair allocation of the road resources. Finally, we ensure reliable data transfers by recovering from vehicles failing to deliver data to the next offloading spot or the final destination. The controller manages the reliability of the data transmission using both redundancy techniques (*e.g.*, RAID level 6) and retransmission mechanisms (*e.g.*, Automatic Repeat reQuest).

In a nutshell, the contributions of this paper are:

- **Centrally controlled vehicular backhaul.** We propose a centralized architecture that enables scalable and adaptive control of the road network to offload traffic.
- **Road resource allocation.** We design an allocation procedure that selects vehicle flows to match the performance requirements of offloading demands.
- **Reliable data transfers.** We combine redundancy and retransmission mechanisms to recover data losses occurring when vehicles fail to deliver the data they transport.
- **Real-world evaluation.** We evaluate our approach for multiple offloading demands assigned on the French road network using actual road traffic counts.

The architecture we propose leverages logical centralization to enable efficient configuration of the road network to offload bulk data transfers. Our results show that data transfers in the order of Petabytes can be offloaded on the roads over distances of several hundreds of kilometers.

The rest of this paper is organized as follows. We give an overview of our offloading system in Section II. In Section III, we motivate the SDN-like architecture and introduce the functions of its components. We then present the vehicle flow allocation procedure with the implementation of the component functions in Section IV. We evaluate the performance of our offloading system with actual traffic counts in Section V. Section VI provides a review of the related work. We then discuss our results and the open issues in Section VII. In Section VIII, we conclude the paper with a summary and give an outlook of future work.

II. Data offloading using offloading spots

The offloading system we propose relies on the use of private vehicles equipped with storage capacities in combination with a collection of fixed wireless data storage devices referred to as *offloading spots* (as illustrated in Figure 1). Vehicles are equipped with one or more removable storage devices such as magnetic disks or other non-volatile solid-state storage devices.

The 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 in charge of supervising the offloaded transfers then charged to the content providers. The service provider also provides incentives to the vehicle owners for the data they transport (*e.g.*, through a "get paid to drive" program). The revenues generated by the offloaded transfers balance the operational costs associated with the deployment of the offloading infrastructure and its maintenance.

We define *data cargo* as the data carried by each vehicle in its storage device. The flow of vehicles so equipped acts as a

mechanical backbone connecting offloading spots. These latter are located where vehicles park for long enough as part of their line of travel, including on-street parking spots, parking lots, or gas stations.

An offloading spot is a fixed device offering short- to medium-term data storage. Data is transloaded from a conventional data network to the road network and stored until shipped to destination by empty vehicles. We take advantage of the parking time to opportunistically load on or unload data from vehicles while stopped at the offloading spots. Offloading spots and vehicles are both equipped with wireless communication interfaces to support short-range radio data exchanges. In the case of electric vehicles, the offloading spots may be located at battery charging stations such as the ones operated worldwide by ChargePoint.[4] In this case, each spot in the station is equipped with wireless transmission devices to allow concurrent transfers when multiple vehicles are parked at the same time.

Offloading spots remove the need of relying on a single vehicle making the trip all the way from the source to the destination of a data transfer. The data is stored until transferred to a subsequent empty vehicle heading toward the intended destination. As a result, data may be transferred to multiple vehicles following different trajectories, thus increasing the utilization of the road resources.

Our offloading system capitalizes on the many segments of trips connecting consecutive pairs of offloading spots. By allowing the data to hop-on and hop-off at any offloading spot along the route between the source and destination, our system is expected to maximize the capacity resulting from the combined storage of all the vehicles. The offloading spots take the decision whether to load data on or off vehicles according to forwarding states installed by an SDN-like controller. The offloading spots act similarly to forwarding engines under the direct authority of the controller. The next section describes the SDN centralized architecture we propose for efficient data offloading onto the road network.

III. SDN CENTRALIZED CONTROL

We first motivate the use of software-defined networking (SDN) and then describe the main components of our vehicle-based offloading architecture as shown in Figure 2. We also fully detail the operation of each component required to offload data traffic on the road network.

A. SDN-enabled road network management

We leverage the advantages of the logical centralization provided by SDN to enable efficient control of the infrastructure to offload bulk delay-tolerant data from a data network. SDN provides the logistics including planning, implementing, and controlling for the effective and efficient transportation of data over the road network [7].

Following SDN's original design, our architecture consists of two components: a central controller and the offloading spots acting as forwarding entities. These components are

[4]http://www.chargepoint.com/

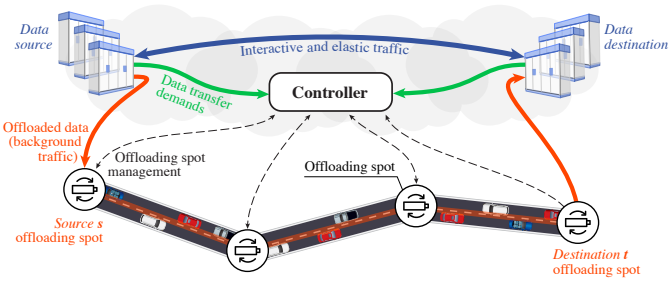

Fig. 2: Centrally-controlled data haulage to offload bulk transfers of delay-tolerant data between two data centers.

depicted in Figure 2. The controller receives the demands to offload data transfers onto the road network. Each demand specifies the volume and delay requirements for the corresponding data transfer. The controller computes the road path consisting of a sequence of offloading spots connected by flows of empty vehicles whose number and speed match the data transfer requirements. The controller connects to the offloading spots and installs the forwarding states needed to select the vehicles that will carry the data to their final destination with respect to the transfer requirements. The controller also configures the scheduling policy that determines in which order to assign data transfers if multiple transfers traverse the same offloading spot.

The functions undertaken by each component are described in the remainder of this section.

B. Controller

The controller maintains a holistic logical view of the offloading infrastructure, including the offloading spots and dynamics such as the traffic volumes on the road paths connecting the offloading spots. It may leverage traffic forecasting techniques such as the ones we present in Section IV-A or services such as Here[5], TomTom[6], or Airsage[7] to characterize the road paths in terms of bandwidth and to update its logical view. The controller uses this logical view to allocate the offloading demands and to make efficient use of the road resources. For reliability purposes, the controller keeps track of the progress of the data transfers at the offloading spots through a low-capacity control channel (*e.g.*, using a cellular network, or long-range technologies such as SigFox[8] or LoRa[9]) [8]. Information about the data transfers includes the cargo waiting to be picked up at offloading spots and the cargo in transit.

The controller also receives statistics about the vehicles parking at offloading spots, including the historical locations of the vehicles made available via the navigation system. The historical locations are stored in a geographic database managed by the controller to help the offloading spots predict the remaining itinerary of the parking vehicles and determine the next offloading spot they are more likely to visit on their

[5]https://www.here.com/business/traffic
[6]http://automotive.tomtom.com/en/connected-services/tomtom-traffic
[7]http://www.airsage.com/Products/Traffic-Insights/
[8]http://www.sigfox.com/
[9]https://www.lora-alliance.org/

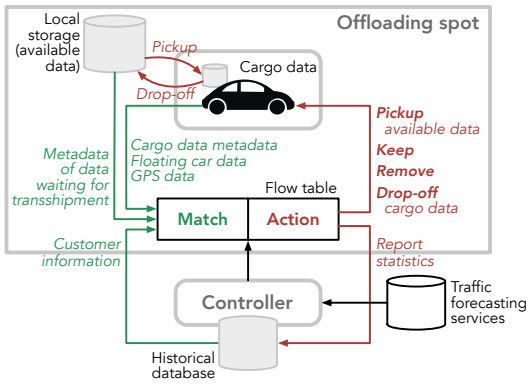

Fig. 3: Forwarding at an offloading spot when a vehicle is parking.

route. In Section VII-B, we review the existing techniques to predict the future direction of the parking vehicles.

C. Offloading demands

The controller receives demands to offload part of or all the data belonging to a transfer on the road network. Each demand specifies the volume and delay requirements for the corresponding data transfer, as well as the entry and exit points. Upon receiving a request to offload a data transfer, the controller computes the optimal road network paths by solving the vehicle flow allocation problem as a multi-commodity flow allocation model (presented in Section IV-C). A road network path consists of a sequence of offloading spots that can accommodate the data transfer requirements. Solving the allocation problem also defines how much data to allocate to the flow of vehicles traveling the stretches of road connecting consecutive offloading spots along the road network paths.

D. Offloading spots

Data is offloaded from a traditional data network to the closest offloading spot using a border dray transfer system. Different techniques to implement such a system are presented in Section VII. Offloading spots are featured with storage capabilities where data is stored until transferred to a parking vehicle via short-range radio. As depicted in Figure 3, subsequent offloading spots act as data relay boxes where the data are dropped off for later pick-up by subsequent empty vehicles. The decision of dropping off or picking up data are dictated by the controller and results from matching the direction of the passing vehicles against the destination of the transfer the data belongs to. As so, the offloading spots act as forwarding engines that select empty vehicles based on their destination to move the offloaded data toward their final destination. Vehicle selection is also driven by the efficient use of the road network resources shared among concurrent offloaded data transfers.

Flow tables. The flow tables determine the forwarding behavior of an offloading spot. They match the direction of the parking vehicles and the destination of the available data cargo with a direction. They consist of a list of entries, each installed for an individual data transfer. The controller adds a new entry in the flow table of the offloading spots located on the road network paths computed for a data transfer. A flow

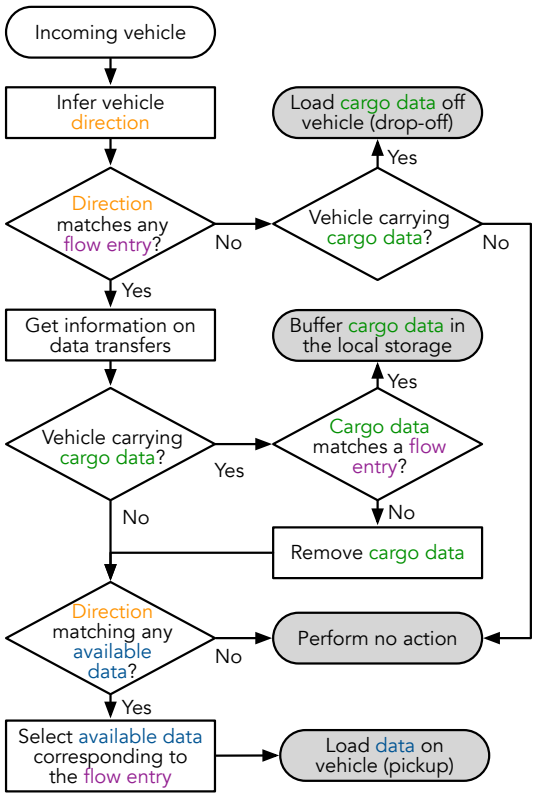

Fig. 4: Forwarding process at an offloading spot.

table entry contains the next hop offloading spot to which the data must be forwarded to reach the destination of the data transfer corresponding to this entry. As depicted in Figure 3, a flow table entry also contains a list of actions to perform on the data. Common actions include loading data on or off the vehicles while parking close to the offloading spot. The controller defines these actions based on the information the offloading spots report on the flows of vehicles.

Forwarding process. The forwarding process determines the data cargo to load on the vehicles parking at the offloading spots from the available cargo waiting to be picked up. It is represented by the flowchart depicted in Figure 4. Upon the arrival of a vehicle, an offloading spot uses the controller to determine the future direction of the vehicle and checks if it matches one entry of its flow table. If none of the entries matches, the vehicle unloads, if any, its data cargo onto the offloading spot storage for future pick-ups and continues its journey without performing any further actions. If multiple entries match the direction of an empty vehicle, the offloading spot selects one entry based on the scheduling strategies presented in Section IV-D. After selecting one of the entries, the offloading spot performs the actions specified in the entry. If the vehicle already carries data, the offloading spot checks if this data belongs to the data transfer represented by the matching entry. If so, the vehicle keeps its cargo and continues its journey. A copy of the cargo is buffered at the offloading spot for reliability purposes. Otherwise, the vehicle unloads its cargo at the offloading spot before resuming its journey. In case the vehicle arrives empty, the offloading spot checks if some data matching the vehicle direction was locally transshipped

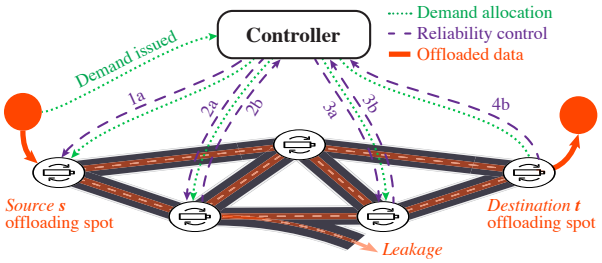

Fig. 5: Controller-to-offloading spot communication for the *hop-by-hop* (dashed arrows 1a-3a and 2b-4b) and *end-to-end* (dashed arrows 1a, 4b) retransmission strategies.

by a previous vehicle or transloaded from the conventional data network. If such data is waiting to be shipped, the data is transferred to the vehicle. Otherwise, the vehicle continues its journey empty loaded.

E. Reliability

The offloading spots keep a copy of the data for later recovery in case the vehicle unexpectedly changes direction before reaching the next offloading spot, runs into an accident, or breaks down. Such events result in data losses that we take into account by introducing a parameter named *data leakage*. To mitigate the effects of data leakage, the controller uses retransmission and redundancy mechanisms.

Retransmissions. Retransmissions of the data cargo completely recover from vehicles failing to deliver the data cargo to the intended offloading spots. We propose a hop-by-hop (hbh) retransmission strategy, as depicted in Figure 5. With this strategy, each offloading spot buffers data for later recovery in case a vehicle fails to deliver its cargo to the next-hop offloading spot. This could result from errors in the prediction of its future direction. In Section VII-B, we review existing techniques to reduce errors in the prediction and to limit the number of retransmissions needed. The controller receives acknowledgments over a feedback channel from the next-hop offloading spot (indicated by dashed arrows 2b, 3b, and 4b in Figure 5) and notifies the previous offloading spot when to retransmit the missing data (indicated by dashed arrows 1a, 2a, and 3a in Figure 5). After a predefined number of attempts, a loss is repaired via the original data network (from where data was first transloaded) to make sure the deadline specified in the offloading demand can be met.

Redundancy. To limit the number of retransmissions, we also use redundancy mechanisms, such as RAID. Redundancy reduces the effect of data leakage by transferring redundant copies of the data, in addition to the original data. The number of redundant copies is determined by the redundancy mechanism (*e.g.*, for one piece of data, two copies are needed with RAID 1, while RAID 6 adds two redundant copies to an array of n data, $n-2$ being available before redundancy). We determine the data overhead resulting from the retransmission and redundancy mechanisms in Section IV-B.

IV. Vehicle Flow Allocation

In this section, we complete the functions of the controller by describing the allocation procedure of the flows of vehicles

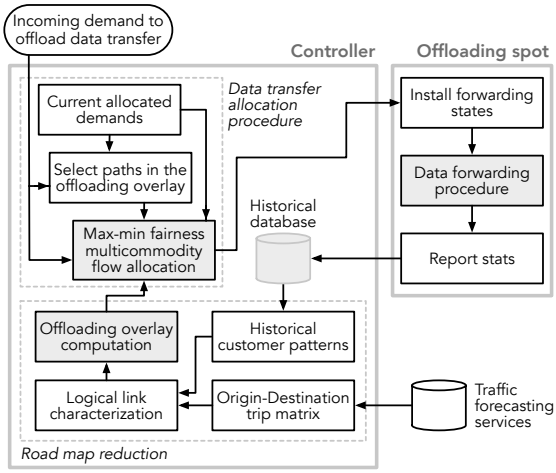

Fig. 6: Interactions between the functions of the controller and those of the offloading spots.

to the offloaded data transfers. A chart of the interactions between the functions of the controller with those of the offloading spots is provided in Figure 6. The allocation procedure involves a reduction of the map complexity we detail in the next section. We then model the allocation of the offloading demands on the road network by accounting for the overhead of the retransmission and redundancy mechanisms that ensure reliable data delivery. We also give more insights into the scheduling policies used by an offloading spot to serve concurrent data transfers.

A. Road map reduction

The controller receives the demands to offload data transfers onto the road network. It assigns then data to vehicles according to their trajectories and also in sufficient number to match the transfer requirements. Prior to the allocation decisions, the controller creates a logical representation resulting from a mapping algorithm. The output of this algorithm is an overlay network that captures the topology of the road network as well as its traffic volumes. Nodes in the overlay network correspond to the offloading spots and are connected through logical links. A logical link corresponds to multiple road paths in the underlying road network. Each logical link is characterized by its capacity, transit time, and data leakage.

The overlay network, termed *offloading overlay*, reduces the complexity of the road network and its high volume of circulating vehicles. The offloading overlay provides the controller with an abstract view of the resources (the flows of vehicles) to be allocated to the offloading demands. This view allows the use of linear programming techniques to solve the, otherwise intractable, problem of allocating the vehicle flows to offloaded data transfers.

In the following, we characterize the logical links of the offloading overlay by expressing the flows of vehicles traveling between adjacent offloading spots in terms of bandwidth. The flows of vehicles are given by an origin-destination matrix between the offloading spots. However, the dataset we use in our simulations only gives the traffic counts in terms of the number of vehicles per unit of time for a set of stretches of road. Therefore, our algorithm leverages traffic forecasting

techniques from transportation research to estimate the origin-destination matrix of the offloading spots from the traffic counts. The algorithm consists of the following four steps.

1) *Route determination.* The first step consists in selecting a subset of the alternative routes 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.* [9].
2) *Route assignment.* The second step consists in assigning weights to the selected routes using the C-logit route assignment model [10]. 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 of all pairs of offloading spots in the offloading overlay [11]. This model determines the most likely distribution of the traffic across all the routes selected in the first step subjected to the traffic counts of the routes' stretches of road and the C-logit weights.
4) *Logical link characterization.* Finally, we determine the attributes of each logical link $(i,\, j)$ in the offloading overlay. These attributes are relevant to the allocation of the vehicle flows. The attributes are as follows:
   - *Capacity* $c(i,\, j)$. The capacity of $(i,\, j)$ represents the combined storage of all vehicles traveling between i and j. The capacity also reflects the market penetration ratio, *i.e.*, the ratio of vehicles equipped with data storage devices.
   - *Travel time* $t(i,\, j)$. The transit 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.

In the rest of this paper, we assume that the capacity of the offloading spots is not limited and that there are no constraints on the number of transfers they can serve. Each vehicle transports a cargo of size σ. An offloading demand d_{st} represents a request to offload a data transfer between source s and destination t. The demand is characterized by the amount of data β_{st} and the deadline τ_{st} before which the transfer should be completed. 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}. We denote by $\mathcal{P}_{st}$ the set of all possible logical paths between the source and destination, respectively s and t. Each logical path $p \in \mathcal{P}_{st}$ consists of a sequence of logical links connecting adjacent offloading spots in the offloading overlay. We list the notations we use in the rest of this paper in Table I.

TABLE I: Table of notations for the vehicle flow allocation problem

Variable	Meaning
$\mathcal{D}$	Set of all offloading demands to allocate
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
σ	Storage capacity of the vehicles
$c(i,\, j)$	Capacity of logical link $(i,\, j)$
$t(i,\, j)$	Transit time on logical link $(i,\, j)$
$l(i,\, j)$	Leakage of logical link $(i,\, j)$
$l_{st}^{\mathbf{red}}(i,\, j)$	Leakage of logical link $(i,\, j)$ with redundancy for offloading demand d_{st}
$o_{st}^{\mathbf{red}}$	Weight of the **red**undancy mechanism on the flow for offloading demand d_{st}
$o_{st}^{\mathbf{ret}}(i,\, j)$	Weight of the **ret**ransmission mechanism on the flow at logical link $(i,\, j)$ for offloading demand d_{st}
$R_{st}(i,\, j)$	Average number of transmissions of a data cargo on logical link $(i,\, j)$ for offloading demand d_{st}
δ_i	data waiting time at offloading spot i
$f(p)$	Flow on logical path p
$t(p)$	Travel time of logical path p
O_{st}	Overhead of the offloading demand d_{st}

B. Reliability overhead

In this section, we express the overhead resulting from the reliability mechanisms we use to mitigate the effects of data leakage, namely redundancy and retransmissions.

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 (equivalent to a disk failure with the typical use of RAID 6). RAID 6 divides the β_{st} data of an offloading demand d_{st} into N arrays of $n \geqslant 4$ 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 6 redundancy increases the data overhead 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 [12]:

$$o_{st}^{\text{red}} = \frac{n}{n-2}. \tag{1}$$

The data carried by n vehicles whose storage devices are arranged in RAID 6 and traveling 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 [12]:

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

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 will explain how n (the number of storage devices per array) is computed 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).

SR-ARQ retransmissions. In addition to data redundancy, we use SR-ARQ (*Selective-Repeat ARQ*) to fully recover the losses that RAID cannot recover [13]. With the hop-by-hop strategy, the controller is informed of loaded vehicles leaving offloading spot i. A copy of the data is stored by i until successfully transmitted to j, the next-hop offloading spot. If no acknowledgment is received from j after $t(i,j)+\varepsilon$ (ε constant), the controller notifies i to retransmit the missing data. In the rest of this section, we will consider a noiseless feedback control channel between the offloading spots and the controller.

The hop-by-hop strategy introduces an overhead corresponding to the average number of transmissions $R_{st}(i,\,j)$ needed to successfully deliver a data cargo on logical link $(i,\,j)$. We express $R_{st}(i,\,j)$ as follows [13]:

$$R_{st}(i,\,j) = \frac{1}{1 - l^{\text{red}}_{st}(i,\,j)}. \tag{3}$$

In the following, we will use $o^{\text{ret}}_{st}(i,\,j) \equiv R_{st}(i,\,j)$ to represent the overhead of the retransmission mechanism over logical link $(i,\,j)$. Note that, while $o^{\text{ret}}_{st}(i,\,j)$ and $R_{st}(i,\,j)$ have the same value, they have different semantics: the former represents an overhead on the allocated flows, whereas the latter represents the number of transmissions on a logical link.

Determination of the array size n. RAID 6 redundancy distributes the data across arrays of n data cargo with n greater than 4. The total number of data cargo n is defined so as to minimize the data overhead O_{st} needed to ensure reliable transfer in response to offloading demand d_{st}:

$$n = \max\left\{4,\, \arg\min_{n}\{O_{st}\}\right\}.$$

The reliability overhead O_{st} for demand d_{st} is expressed as the summation of the retransmission overhead $o^{\text{ret}}_{st}(i,\,j)$ of the logical links $(i,\,j)$ followed by the data cargo, weighted by the redundancy overhead o^{red}_{st}:

$$O_{st} = o^{\text{red}}_{st} \sum_{\substack{p\in\mathcal{P}_{st}\\(i,\,j)\in p}} o^{\text{ret}}_{st}(i,\,j).$$

We propose to determine the optimal value of n by computing the resulting overhead, for each value of n in $[4,\,50]$. The optimal value of n is the one that minimizes the resulting overhead O_{st} for demand d_{st}. The larger the leakage on logical links $(i,\,j)$, the larger the resulting retransmission overhead $o^{\text{ret}}_{st}(i,\,j)$, and the smaller n, as the redundancy must take account of the additional retransmissions.

C. Vehicle flow allocation procedure

The controller receives the demands to offload data transfers characterized by their performance requirements. The task of the controller consists in selecting the empty vehicles traveling in the direction of the transfer destinations such that (*i*) the number of vehicles is sufficient to meet the transfer requirements and (*ii*) the allocation of the vehicles' combined storage is efficient and fair among the competing transfers. To this end, the controller starts by computing the logical paths consisting of a sequence of logical links selected according to their properties as specified in the offloading overlay. The controller then configures the offloading spots along the selected logical paths.

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. The travel time $t(p)$ experienced by the cargo assigned to successive vehicles over logical path p is determined by the sum of two components: the transit component and the waiting component. The transit component is the sum of the transit time of each logical link in p. The waiting component is the sum of the waiting times experienced at each offloading spot connecting those logical links. We express $t(p)$ as a function of $R_{st}(i,j)$, the average number of transmissions to successfully deliver a cargo from i to j for demand d_{st}, as follows:

$$t(p) = \sum_{\substack{p\in\mathcal{P}_{st}\\(i,\,j)\in p}} R_{st}(i,\,j)[\delta_i + t(i,\,j)], \tag{4}$$

where δ_i is the waiting time at offloading spot i, large enough to transfer σ data between the vehicle and the offloading spot.

Linear programming formulation. We formulate the vehicle flow allocation procedure as a linear programming (LP) model that determines the logical paths matching the performance requirements of the offloading demands. The LP shown in Figure 7 consists in allocating $f(p)$ flows of data on the vehicles traveling the logical paths listed in $\mathcal{P}_{st}$. We first present the inputs and then the allocation strategy we use in the allocation procedure. The strategy relies on the multi-commodity flow allocation problem we formulate as a linear programming model.

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

Procedure. The controller allocates $f(p)$ flows to the logical paths of $\mathcal{P}_{st}$ for each demand and according to the Max-Min fairness strategy. The Max-Min fairness 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 following iterations successively allocate the remaining capacity of the network to the demands that can receive more flows. More specifically,

Input: Offloading demands $\mathcal{D} = \{d_{st}\}$, each between s and t characterized by β_{st} and τ_{st}
Set $\mathcal{P}_{st}$ of possible logical paths between s and t
Average transit time $t(p)$ on logical path p
Capacity $c(i,\, j)$ of logical link $(i,\, j)$
Output: Flow allocation $A = \{f(p)\}$ 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 demands 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 ϕ_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 $\{\phi_i\} \cup \{f(p) : p \in \mathcal{P}_{st},\, (s,\, t) \in \mathcal{D}\}$

Fig. 7: Vehicle flow allocation procedure.

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 following iterations process the remaining demands. To determine for which transfers the current allocation can be further increased, we use the non-blocking test algorithm presented in Figure 8.

The core of the Max-Min fairness algorithm is the MCF function shown in Figure 7. 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, $k \in [0,\, i-1]$ keep the same allocation they received at previous steps 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 fixed by the MCF function for iteration i, the Max-Min fairness algorithm determines which

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 commodities 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 t_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:

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

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

Fig. 8: Non-blocking test for the Max-Min fairness allocation procedure.

demands can be further increased in their current allocation using the non-blocking test algorithm shown in Figure 8. The non-blocking test is derived from the algorithm proposed by Pióro *et al.* [14]. 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 is non-blocking and 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. In case the requirements of a demand are satisfied, the flows allocated to this demand can be further increased because of the first constraint of the multi-commodity flow function in Figures 7 and 8. As a result, the transfer will be completed before the deadline provided in the demand.

For backlogged demands, the amount of data is infinite (*i.e.*, $\beta_{st} = \infty$). As a result, the demand rate is also infinite (*i.e.*, $\lambda_{st} = \infty$). In order for the above formulation of the multi-commodity flow to remain valid, we need to ignore the first constraint in the MCF function that satisfies the demand requirements.

If the ratio β_{st}/τ_{st} of a demand d_{st} is larger than the aggregated flows of vehicles $\sum_{p \in \mathcal{P}_{st}} f(p)$ traveling the logical paths p between s and t, the demand cannot be allocated. In this case, the multi-commodity flow allocation does not give any solutions. As a result, an access control policy could make sure that the demand requirements are compatible with the capacity offered by the system. If the demand is not feasible, the policy can negotiate a larger deadline or a smaller amount of data to transfer.

D. Data scheduling

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 which data cargo to load on the stopping vehicle according to the weighted fair queuing scheduling policy 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 of offloading demand d_k. The controller assigns a weight $w(d_k, p_l)$ to the transfers allocated to the logical paths in $P_{i,j} = \{(d_k, p_l) \mid \forall d_k \in \mathcal{D}, p_l \in \mathcal{P}_{d_k}, (i, j) \in p_l\}$. $P_{i,j}$ 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 $P_{i,j}$:

$$w(d_k, p_l) = \frac{f(d_k, p_l)}{\sum_{p \in P_{i,j}} f(p)}. \quad (5)$$

The weights are used with a scheduling algorithm to determine in which order to assign the cargo to a passing vehicle if multiple transfers traverse the same offloading spot. In our simulations, we considered a weighted round robin scheduler [15] and a probabilistic weighted fair queuing scheduler [16]. While they both have the same throughput performance, we found that the former better distributes the data amongst vehicles. The weighted round robin scheduler helps overcome bufferbloat and improves the end-to-end delay performance.

V. Evaluation on the French road network

In this Section, subsections V-A through V-B describe the setup and subsections V-C through V-E present the results. The objective of the simulation is to evaluate three metrics: (*i*) maximum throughput to evaluate the capacity of our offloading system, (*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 consider the allocation procedure in the context of a network of charging stations for electric vehicles deployed across France. Data is loaded on and off the vehicles while charging their batteries. We evaluate the performance resulting from the different allocation strategies presented in Section IV-C. The maximum throughput is expressed as the number of data cargo of an arbitrary size delivered per second and the end-to-end delay is for a data cargo size of $\sigma = 1$ Pb. In the rest of this section, we consider a conservative market penetration ratio of 10%. The market penetration ratio represents the share of vehicles equipped with storage capabilities and ready to participate in the data offloading. We also set the waiting time $\delta_i = 30$ minutes at each offloading spot i. This waiting time corresponds to the time needed to provide up to 300 km of range when an EV is charging its battery.[10] This time allows

[10]https://www.tesla.com/supercharger

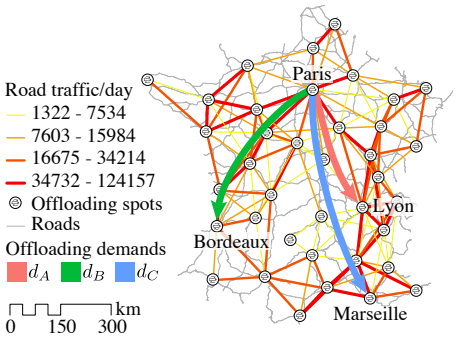

Fig. 9: Deployment plan of charging stations and offloading overlay representing the network of charging stations for electric vehicles deployed to cover the roads of France. The thickness of the lines are proportional to the computed capacity of the logical links.

1 Tb transfers using state-of-the-art high-throughput wireless technology (*e.g.*, MIMO 802.11ac) between the EV and the charging station [17].

A. French highway dataset

We implement a realistic deployment plan of charging stations covering all of France as depicted in Figure 9. The charging stations are located 150 km apart and their placement is determined by solving a facility allocation problem [18]. The resulting network of charging stations helps extend the driving range of the electric vehicles, while minimizing the number of charging stations. We feed the road map reduction algorithm presented in Section IV-A with actual traffic counts provided by the AADT (*Annual Average Daily Traffic*) of the major roads in France covering a combined distance of 20,000 km.[11] The resulting offloading overlay is shown in Figure 9.

B. Vehicle flow allocation

We evaluate the performance of the transfers resulting from the allocation of three offloading demands on top the offloading network consisting of charging stations deployed in France as described above. The three demands are shown in Figure 9: (*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 [19] to simulate microscopic vehicular traffic and run our 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 destination. Data leakage is assumed to be the same for all logical links in the offloading overlay with a default value of 30% unless otherwise noted. We considered the same simulation run for each of the experiments by recording the events when a vehicle arrives to and departs from an offloading spot.

[11]Census of the road traffic on the French road network in 2011 (in French): http://tinyurl.com/otfbewv

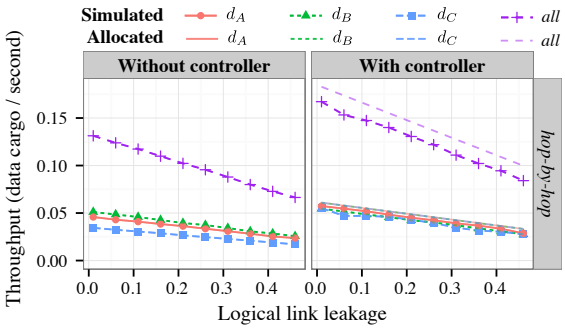

Fig. 10: The maximum throughput achieved for offloading demands d_A, d_B and d_C (depicted in Figure 9) as a function of the logical link data leakage (assumed to be the same on all the logical links).

C. Maximum throughput

To assess the need of 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 passing by each offloading spot. The strategy with a controller consists in selecting the data transfers in a round-robin order locally at each offloading spot, while the strategy with a controller relies on the Max-Min fairness allocation presented in Section IV-C. For both strategies, we evaluate the maximum throughput achieved by each transfer d_A, d_B, and d_C.

To evaluate the maximum throughput that the system can achieve, we consider an infinite backlog traffic generated at the data sources (placed in Paris) for each demand (*i.e.*, $\lambda_A = \lambda_B = \lambda_C = \infty$). We evaluate the maximum throughput for each strategy and the hop-by-hop retransmission mechanism introduced in Section III-E.

We plot the maximum throughput in Figure 10 as a function of the data leakage for the hop-by-hop retransmission mechanism. The maximum throughput achieved for each of the transfers is expressed in terms of data cargo delivered per second. We represent the results of the simulations performed with SUMO following the Max-Min fairness allocation procedure with a controller.

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 9, d_A and d_C compete for the same resources, as they both follow road paths sharing common logical links. The strategy without a controller allocates the flow of vehicles traveling those links to the respective destinations of d_A and d_C without taking into account that destination of demand d_C is farther away compared to demand d_A. Thus, this 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.

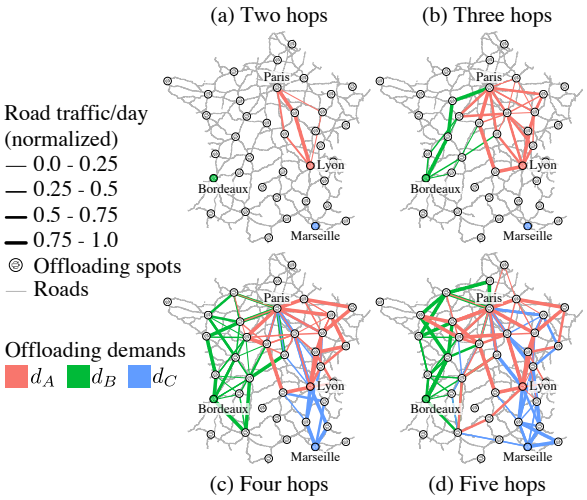

Fig. 11: Representation of the allocation of demands d_A, d_B, and d_C resulting of the strategy with controller with different values for the maximal length of the candidate logical paths.

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 that the controller allocates the flows of vehicles by solving the Max-Min fairness allocation presented in Section IV-C. The higher performance of the strategy with a controller further confirms its need compared to an architecture without a controller.

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

D. Number of offloading spots per data transfers

In a second step, we examine the impact of the number of offloading spots on the duration needed to satisfy demands d_A, d_B, and d_C. We now consider that each demand requests a transfer of 10 PB of data without specifying any deadline, and each vehicle can transport a data cargo of size σ =1 Tb. 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 12. 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 by the number of offloading spots. We also measure the mean travel time of a 1 Tb cargo which corresponds to the cargo size of a vehicle. 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.

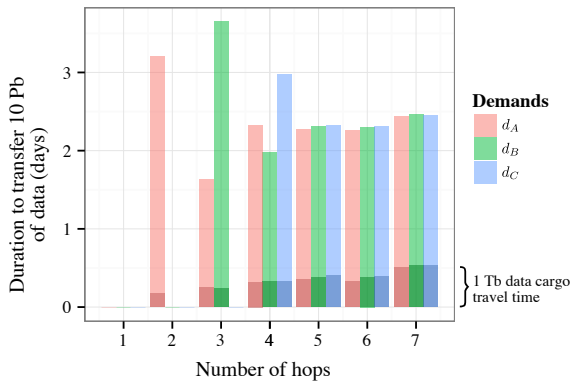

Fig. 12: The duration needed to complete a 10 Pb transfer (in lighter colors) and the travel time of a 1 Tb data cargo (in darker colors) as a function of the candidate path maximal length in terms of hops.

We examine the results in Figure 12 together with Figure 11 where we represent the logical paths allocated for each demand depending on the maximal length of the candidate paths.

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 11a. 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 11b 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 11c. 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 11c. 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 1 Tb cargo. Note that for paths of five hops and more, a deadline $\tau_{st} = 3$ days is sufficient to transfer the data of each demand.

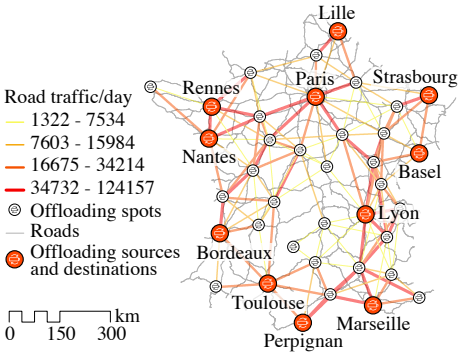

Fig. 13: Traffic matrix to be allocated among the cities represented on the map.

E. Complete traffic matrix

We increase the stress on our system by allocating concurrent demands, all issued among the eleven cities represented in Figure 13. We use the strategy with a controller to allocate the demands at the same time and consider again the hop-by-hop retransmission mechanism, given a 30% data leakage for all logical links.

This results in a total of 110 concurrent demands to allocate on the French roads. We represent in Figure 14 the throughput resulting from the Max-Min Fairness allocation procedure for each transfer, expressed as the number of cargo data delivered per second times 1,000. We first note that the minimum throughput allocated to the demands is 0.0038 data cargo per second, which is equivalent to 3.8 Gbps for data cargo of size $\sigma = 1$ Tb. Secondly, the total aggregated throughput is equal to 0.822 data cargo per second, which is almost three times greater than the aggregated throughput of the experiment presented in Section V-E (compared to 0.11 data cargo per second). Since the demands cover a larger area of the road network, they get more capacity than the demands of the previous section. Thirdly, we notice that some demands receive significantly more throughput than other demands. This is the case of the following demands, listed as source-destination pairs: (Marseille, Lyon), (Paris, Nantes), (Rennes, Paris), and (Rennes, Nantes). These demands benefit from high-capacity logical paths of a few hops that correspond large highways in the road network.

VI. Related Work

A. Delay-Tolerant Networking

Our work shares some features with the paradigm of delay tolerant networks (DTNs), which leverages on the mobility of a wide range of entities including vehicles. The main focus of the research on DTN has been on routing in sparse, partially connected networks operating in challenging environments where low node density and lack of infrastructure have motivated the introduction of the so-called store, carry, and forward principle. Data is transported by mobile nodes and passed on hop-by-hop asynchronously when a node encounters another peer. Encounters between nodes are seen as opportunities for

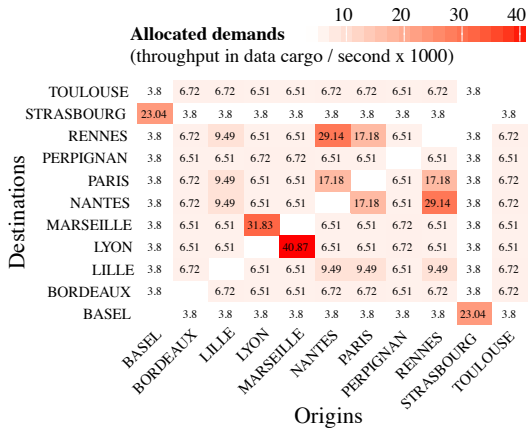

Fig. 14: Traffic matrix of the cities depicted in Figure 13 with the throughput of allocated demands expressed in terms of data cargo per second $\times$ 1000.

forwarding data until eventual delivery to remote locations with poor or non-existing connections.

DakNet [20] proposes to bring asynchronous Internet connectivity to rural villages by relying on *mobile access points* (MAP) consisting of portable storage devices mounted on buses, motorcycles, or bicycles acting as data carriers between kiosks and Internet access points called hubs. A MAP exchanges data whenever in the vicinity of a kiosk where data is collected from or delivered to residents in the surrounding area. Whenever the MAP comes within range of a hub, data is then uploaded to or downloaded from the Internet. An example of real deployment of such network is the UMass DieselNet testbed [21]. DieselNet is a DTN network consisting of 40 buses equipped with WiFi capabilities serving the surrounding area of UMass Amherst campus. The testbed was used to measure and to model the intermittent connectivity between buses, as well as to evaluate MaxProp, a routing protocol for delay-tolerant networks. In the case of MaxProp, the data forwarding results from local decisions made by each bus according to a *Delivery Likelihood* estimation based on history information about past meetings with other buses.

The MULE (*Mobile Ubiquitous LAN Extensions*) architecture aims to provide connectivity within sparse sensor networks by exploiting the random mobility of humans, animals, and vehicles [22]. The MULE architecture enables opportunistic collection of sensed data from the source sensors to central repositories for analysis purposes. Instead of relying on random node mobility, Zhao *et al.* proposed the use of small, inexpensive battery-powered devices equipped with storage and wireless interfaces called *throwboxes* [23]. Throwboxes enhance the delivery likelihood of DTN networks by increasing the contact opportunities when placed at strategic locations. A mobile node can offload its messages to the closest throwbox where it will be stored until transmitted later to another mobile node. The forwarding decision which amounts to transmit a message to and from a throwbox requires the knowledge of the node mobility.

In a companion work, Zhao *et al.* considered special mobile nodes, called *message ferries* acting as relays in charge of carrying data between disconnected nodes [24]. The mobility of the message ferries is controlled so they get closer to other nodes and enhance the data delivery ratio.

While most research achievements on DTNs have focused on the forwarding strategies or the prediction of node mobility to enhance the delivery success in sparse networks, our work leverages the increasing number of *private* vehicles and miles traveled to overcome the capacity limitation of conventional data networks such as the Internet by offloading large data cargo on the road network.

B. Offloading techniques

Offloading techniques aim at using alternative communication channels to relieve and assist conventional data networks from large amounts of data. These techniques were originally proposed in the context of cellular networks such as 3G/4G LTE, to use complementary wireless communications such as WiFi to access the Internet [25]. There are two ways to carry out the offloading. The first is to use fixed WiFi access points to provide additional and efficient connectivity to mobile users. This technique is used by major carriers with deployed access points to relieve their cellular networks [26]. In the context of vehicular Internet access, Balasubramanian *et al.* proposed to study the feasibility of data offloading from 3G base stations using these WiFi access points [27]. The second is to use direct communications between mobile users, thus creating a delay-tolerant network. The data delivered from the cellular base stations to a small fraction of mobile users, either humans or vehicles, that will in turn disseminate the data as they move and encounter other users [28], [29].

While these offloading techniques aim to provide an alternative access to the Internet to mobile users, our system creates an alternative communication channel from the movements of private vehicles carrying data. In our use case, we use this channel to deliver massive amounts of data between remote locations. Related offloading approaches propose to exploit the mobility of airline passengers traveling on scheduled flights to transfer data between airports [30]. In this approach, data is loaded on passenger phones before take-off and unloaded when they land at their destination. Another proposal suggests shipping data stored on hard drives packed in parcels using the postal service [31]. Contrary to the aforementioned vehicle-based delay tolerant networks, those proposals use mobile entities *dedicated* to passenger transportation or parcel delivery. They exhibit thus deterministic and predictable mobility patterns resulting from the predefined schedules they follow. In our work, the control and management of a vehicular offloading system are made more challenging due to the non-deterministic mobility of the private vehicles we exploit along their line of travel.

C. Transportation research

An alternative solution to our offloading system consists in operating a fleet of trucks dedicated to the door-to-door transportation of data. To make up for the restricted size of the fleet in comparison to the number of private cars being operated on the roads, trucks should be equipped with higher capacity data storage and higher data-rate wireless communication capabilities compared. An offloading service

provider may be in charge of planning the delivery routes performed by the fleet of trucks so equipped. The allocation of the resources (in this case, the dedicated trucks) to transport the data parcels refers to the transportation problem [32], a common problem in transportation research. If data needs to be shipped across a sequence of intermediate transport hubs before reaching final destination, the problem refers then to the transshipment problem [33], also a very common problem in transportation research.

VII. Further implementation issues

A. Transloading phase

As shown in Figure 1, the data have to be transloaded, that is transferred both from its source (*e.g.*, a data center) to the first edge offloading spot of the transfer and from the last edge offloading spot to the data destination (*e.g.*, a remote data center). Note that the transloading is equivalent to the first and last mile of the access networks in an Internet-based transfer. The transloading can be carried out by different means:

- *Dedicated lines* (*e.g.*, optical fiber channels) can be set up between the different locations that specifically need transloading. This solution seems adapted to connect locations that require continuous large data transfers. Although, it may not fit temporary data transfers, as it is a costly solution (*e.g.*, the sources and destinations of a data transfer are only temporary).
- *Dedicated vehicles* may provide transloading 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 private vehicles we consider in the paper.

B. Prediction of a vehicle's direction

The forwarding process at the offloading spots relies on the itinerary prediction of the stopped vehicles. The itinerary gives the next stop the vehicle will make at another offloading spot. The current offloading then uses this information to select the available data cargo to load in the vehicle's storage. Most of the previous work on modeling and predicting transportation routines rely on the location history of the drivers: drivers often go where they have been before. In our offloading system, we can leverage the central controller that gather information from the offloading spots.

Some vehicles may be equipped with a route planner device programmed to give an itinerary to the destination intended by the driver. The drivers may share their planned itinerary with the offloading spot, which includes the planned route and intended destination of the vehicle.

Historical databases managed by the controller log the stops the participating vehicles made at the offloading spots. The future route of the vehicle can be predicted by knowing partial trajectories of the vehicles, using probabilistic tools, such as Hidden Markov Models [34], maximum entropy [35], or Bayesian networks [36], [37]. The partial trajectories of the vehicles can be known through the successive locations recorded by the route planner device.

The current road traffic in the vicinity of the offloading spot can help predict the most likely routes vehicles will take [38]. The offloading spot offloads the information collected on the vehicles to the controller. The controller then infers the next stop the vehicle will most likely make and determines the traffic flow to which the vehicle belongs. The decision is transmitted to the offloading spot, which selects the data cargo to load on the vehicle. The selection follows the scheduling algorithm derived from the installed forwarding states.

VIII. Conclusion and perspectives

In this paper, we take advantage of the bandwidth resulting from the mobility of private vehicles equipped with storage capabilities to offload massive amounts of delay-tolerant traffic from the Internet. We propose an SDN-based architecture consisting of a controller and a collection of fixed wireless data storage devices called offloading spots acting as forwarding engines. The controller receives the demands to offload all or part of a data transfer and selects the flows of vehicles connecting a sequence of offloading spots that match the transfer performance requirements in terms of bandwidth and latency. The controller computes the sequence of offloading spots by solving the vehicle flow allocation problem with a Max-Min fairness allocation and connects to those offloading spots to install the forwarding states and configures the scheduling strategy.

We leverage on the advantages of the logical centralization provided by SDN to alleviate the complexity of the road network topology and the large number of vehicular trips. SDN allows flexible and scalable configuration of the offloading infrastructure for efficient and fair allocation of the vehicles' combined storage among the competing transfers. We evaluate our approach with simulated road traffic for multiple offloading demands assigned on the French road network using actual road traffic counts. With only 10% of vehicles equipped with 1 Tb of storage, our results show that several Petabyte of data can be offloaded in a single transfer covering several hundreds of kilometers, while delivered in less than a day.

As future work, we plan to extend our architecture by transferring the forwarding capabilities of the offloading spots to the vehicles, as data can be exchanged without requiring stationary data relays. We also intend to equip vehicles with sensing and processing capabilities, as they can be turned into mobile sensors in the context of smart cities and the Internet of things.

References

[1] B. Baron, P. Spathis, H. Rivano, M. D. De Amorim *et al.*, "Vehicles as big data carriers: Road map space reduction and efficient data assignment," in *IEEE VTC-Fall*, Vancouver, Canada, Sep. 2014.

[2] B. Baron, P. Spathis, H. Rivano, and M. D. de Amorim, "Offloading massive data onto passenger vehicles: Topology simplification and traffic assignment," *to appear in IEEE/ACM Transactions on Networking*, 2016.

[3] Cisco Visual Networking Index (VNI), "Forecast and methodology, 2015-2020," Jun. 2016.

[4] J. Hecht, "The bandwidth bottleneck that is throttling the Internet," *Nature*, vol. 536, no. 7615, p. 139, Aug. 2016.

[5] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: a platform for high-performance internet applications," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, 2010.

[6] N. Laoutaris, G. Smaragdakis, R. Stanojevic, P. Rodriguez, and R. Sundaram, "Delay-tolerant bulk data transfers on the Internet," *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 1852–1865, Jan. 2013.
[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," in *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, Mar. 2008, pp. 69–74.
[8] M. Anteur, V. Deslandes, N. Thomas, and A.-L. Beylot, "Ultra narrow band technique for low power wide area communications," in *IEEE GLOBECOM*, San Diego, CA, USA, Dec. 2015.
[9] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck, "Alternative routes in road networks," *ACM JEA*, vol. 18, pp. 1.3:1–17, Apr. 2013.
[10] E. Cascetta and A. 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, Jul. 1996.
[11] H. J. Van Zuylen and L. G. Willumsen, "The most likely trip matrix estimated from traffic counts," *Elsevier Transportation Research Part B: Methodological*, vol. 14, no. 3, pp. 281 – 293, Sep. 1980.
[12] P. Chen, E. Lee, and G. Gibson, "RAID: High-Performance, Reliable Secondary Storage," *ACM Computing*, 1994.
[13] S. Lin and D. J. Costello, *Error control coding*. Prentice-hall Englewood Cliffs, 2004.
[14] M. Pióro, P. Nilsson, E. Kubilinskas, and G. Fodor, "On efficient max-min fair routing algorithms," in *IEEE ISCC*, Kemer, Turkey, Jun. 2003.
[15] M. H. Ammar and J. W. Wong, "The design of teletext broadcast cycles," *Elsevier Performance Evaluation*, vol. 5, no. 4, pp. 235–242, 1985.
[16] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 375–385, 1996.
[17] R. Van Nee, "Breaking the gigabit-per-second barrier with 802.11 ac," *IEEE Wireless Communications*, 2011.
[18] C. Toregas, R. Swain, C. ReVelle, and L. Bergman, "The location of emergency service facilities," *INFORMS Operations Research*, vol. 19, no. 6, pp. 1363–1373, 1971.
[19] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo-simulation of urban mobility-an overview," in *SIMUL*, Barcelona, Spain, Oct. 2011.
[20] A. Pentland, R. Fletcher, and A. Hasson, "Daknet: Rethinking connectivity in developing nations," *IEEE Computer*, vol. 37, no. 1, pp. 78–83, Jan. 2004.
[21] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for vehicle-based disruption-tolerant networks," in *IEEE INFOCOM*, Barcelona, Spain, Apr. 2006.
[22] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks," *Elsevier Ad Hoc Networks*, vol. 1, no. 2, pp. 215–233, Sep. 2003.
[23] W. Zhao, Y. Chen, M. Ammar, M. Corner, B. Levine, and E. Zegura, "Capacity enhancement using throwboxes in DTNs," in *IEEE MASS*, Vancouver, BC, Canada, Oct. 2006.
[24] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *ACM MobiHoc*, Tokyo, Japan, May 2004.
[25] F. Rebecchi, M. D. De Amorim, V. Conan, A. Passarella, R. Bruno, and M. Conti, "Data offloading techniques in cellular networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 580–603, 2015.
[26] J. Research, "Data Offload Connecting Intelligently," White Paper, 2013.
[27] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3g using wifi," in *ACM Mobisys*, San Francisco, CA, USA, Jun. 2010.
[28] Y. Li, G. Su, P. Hui, D. Jin, L. Su, and L. Zeng, "Multiple mobile data offloading through delay tolerant networks," in *ACM International Workshop on Challenged Networks (CHANTS)*, Las Vegas, NV, USA, Sep. 2011.
[29] F. Rebecchi, M. D. de Amorim, and V. Conan, "Droid: Adapting to individual mobility pays off in mobile data offloading," in *IEEE/IFIP Networking Conference*, Trondheim, Norway, Jun. 2014.
[30] A. Keränen and J. Ott, "DTN over aerial carriers," in *ACM CHANTS*, Beijing, China, 2009.
[31] R. Y. Wang, S. Sobti, N. Garg, E. Ziskind, J. Lai, and A. Krishnamurthy, "Turning the postal system into a generic digital communication mechanism," in *ACM SIGCOMM Computer Communication Review*, vol. 34, 2004, pp. 159–166.
[32] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial & Applied Mathematics*, vol. 5, no. 1, pp. 32–38, Mar. 1957.
[33] B. Hoppe and É. Tardos, "The quickest transshipment problem," *INFORMS Mathematics of Operations Research*, vol. 25, no. 1, pp. 36–62, Feb. 2000.
[34] R. Simmons, B. Browning, Y. Zhang, and V. Sadekar, "Learning to predict driver route and destination intent," in *IEEE ITSC*, Toronto, ON, Canada, Sep. 2006.
[35] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI*, Chicago, IL, USA, Jul. 2008.
[36] L. Liao, D. J. Patterson, D. Fox, and H. Kautz, "Learning and inferring transportation routines," *Elsevier Artificial Intelligence*, vol. 171, no. 5, pp. 311–331, 2007.
[37] J. Krumm and E. Horvitz, "Predestination: Inferring destinations from partial trajectories," in *Springer UbiComp 2006: Ubiquitous Computing*, Orange County, CA, USA, Sep. 2006.
[38] G. Xue, Z. Li, H. Zhu, and Y. Liu, "Traffic-known urban vehicular route prediction based on partial mobility patterns," in *IEEE ICPADS*, Shenzhen, China, Dec. 2009.

Benjamin Baron is a researcher working at UPMC Sorbonne Universités. He received his Ph.D. in Computer Science from UPMC in 2016. His research interests include the study and design of delay-tolerant networks in mobile environments.

Prométhée Spathis is an associate professor at UPMC Sorbonne Universités since 2005. He received his Ph.D. in Computer Science from UPMC in 2003. His main research interest is in the study of data kinematics in large-scale mobile systems.

Hervé Rivano is a senior INRIA researcher. He is the head of the Inria Agora team in the Inria/INSA Lyon CITI laboratory. He graduated from Ecole Normale Supérieure de Lyon in 2000 and got his Ph.D. from University of Nice Sophia Antipolis in 2003. His research interests include combinatorial optimization applied to Smart Cities networks design and provisioning.

Marcelo Dias de Amorim (`http://www-npa.lip6.fr/~amorim`) is a CNRS Research Director at the LIP6 Computer Science laboratory from UPMC Sorbonne Universités. His research interests focus on mobile networked systems.

Yannis Viniotis received his Ph.D. from the University of Maryland, College Park, in 1988 and is currently a Professor with the Department of Electrical and Computer Engineering at North Carolina State University. Dr. Viniotis is the author of over one hundred technical publications, including two engineering textbooks. He has served as the cochair of two international conferences in computer networking. His research interests include virtualization, service engineering, IoT and design and analysis of stochastic algorithms as they apply to network management. Dr. Viniotis was the cofounder of Orologic, a successful startup networking company in Research Triangle Park, NC, that specialized in ASIC implementation of integrated traffic management solutions for high-speed networks.

Mostafa H. Ammar (F02) received the S.B. and S.M. degrees from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1978 and 1980, respectively, and the Ph.D. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1985. He is a Regents Professor with the College of Computing, Georgia Institute of Technology (Georgia Tech), Atlanta, GA, USA. He has been with Georgia Tech since 1985. His research interests are network architectures, protocols, and services. He has contributions in the areas of multicast communication and services, multimedia streaming, content distribution networks, network simulation, disruption tolerant networks, mobile cloud computing and network virtualization. He has published extensively in these areas and, to date, graduated 35 PhD students. He served as the Editor-in-Chief of the IEEE/ACM TRANSACTIONS ON NETWORKING from 1999 to 2003. He is a Fellow of the ACM.