Cooperative download in urban vehicular networks

Marco Fiore
Université de Lyon
INRIA, INSA-Lyon, CITI
F-69621, France

Jose M. Barcelo-Ordinas
Departament d'Arquitectura de Computadors
Universitat Politècnica de Catalunya
Barcelona, Spain

***Abstract*—We target urban scenarios where vehicular users can download large files from road-side Access Points (APs), and define a framework to exploit opportunistic encounters between mobile nodes to increase their transfer rate. We first devise a technique for APs deployment, based on vehicular traffic flows analysis, which fosters cooperative download. Then, we propose and evaluate different algorithms for carriers selection and chunk scheduling in carry&forward data transfers. Results obtained under realistic road topology and vehicular mobility conditions show that coupling our APs deployment scheme with probabilistic carriers selection and redundant chunk scheduling yields a worst-case 2x gain in the average download rate with respect to direct download, as well as a 10x reduction in the rate of undelivered chunks with respect to a blind carry&forward.**

I. Introduction

Vehicular environments are one of the most promising fields of application for pervasive mobile networking. If the fast dynamics of the topology of a network built over moving vehicles make the adoption of a fully ad-hoc paradigm unfeasible, schemes relying on vehicle-to-infrastructure and opportunistic car-to-car communication appear instead viable. Among the possible applications of infrastructure-based opportunistic vehicular networks, we focus on that of cooperative download.

In fact, although having recently drawn increasing attention from the research community, studies on vehicular cooperative download have been limited to highway scenarios, where the unidimensional nature of nodes mobility simplifies the problem. In this paper, we move for the first time vehicular cooperative download to urban environments.

We consider an urban scenario, where users aboard cars are interested in downloading large files from servers in the Internet, exploiting to this end APs located over the road topology. Obviously, not all users download large files all the time: indeed, one can expect that the portion of vehicles actively downloading large contents is small over the entire car traffic, as it also happens in traditional networks [1]. Therefore, at each time instant, only a very few APs are involved in data transfers to downloader cars in their coverage area, and the majority of APs is instead idle.

Our goal is to exploit these APs inactivity periods to transmit, to cars currently within range of idle APs, pieces of the data other vehicles in the network are interested in. Cars that obtain information chunks this way can then transport the data according to a *carry&forward* paradigm [2], and deliver it to the destination vehicle, exploiting opportunistic contacts with it. An example of this procedure is provided in Fig. 1.

This work has been supported by the EuroNF NoE and by the Spanish Ministery of Science and Technology under grant TSI2007-66869-C02-01.

978-1-4244-5113-5/09/$25.00 ©2009 IEEE

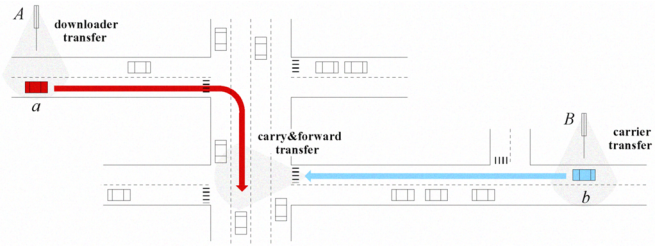


Fig. 1. Example of cooperative download in an urban environment. Vehicle a downloads part of some content from AP A, through a direct link. The idle AP B then gives another portion of the same content to a local vehicle b. When b encounters a, vehicle-to-vehicle communication is employed to transfer to a the data carried by b

Such cooperative download process becomes especially challenging in urban environments, where the presence of multiple route choices makes it hard to predict if vehicles will meet, while the interaction among individual cars under precise driving rules render contacts timings very variable.

In this paper, we identify and address three main challenges:

- *APs deployment*: urban roads are not all identical, as some are more congested than others, some are bidirectional and others one-way, some have higher speed limits than others. This must be taken into account when deploying APs, since diverse plannings of the infrastructure can yield dramatic differences in terms of download rate achieved by vehicles. APs deployment techniques must be thus devised to favor the cooperative download process among vehicles;
- *carriers selection*: contacts between cars in urban environments are not easily predictable like in highway scenarios. Idle APs cannot randomly or inaccurately select vehicles to carry&forward data, as most of the chunks risk to be never delivered to their destinations. Choosing the right carriers for the right downloader vehicles is thus a key issue in urban cooperative download;
- *chunk scheduling*: selecting which parts of the information should be assigned to carriers, and in particular choosing the level of redundancy in this assignment, plays a major role in reducing chunk losses in the system.

Solutions to these problems are among the original contributions of the paper, which include: (i) an APs deployment scheme based on analysis of vehicular flows and optimization of *crossing volumes*, described in Sec. II; (ii) three carriers selection algorithms, based on vehicular contact probability estimation, introduced in Sec. III; (iii) three chunk scheduling techniques, characterized by different redundancy levels, detailed in Sec. IV; (iv) the first performance evaluation of cooperative download in urban scenarios, conducted in realistic

vehicular mobility conditions over real road topologies, whose results are presented in Sec. V. Related work is discussed in Sec. VI, and conclusions are drawn in Sec. VII.

II. Access Points Deployment

A judicious placement of APs over the urban road topology is a fundamental first step in the definition of an efficient cooperative download architecture. In particular, we target a deployment of APs that maximizes the potential for collaboration among vehicles, so to favor carry&forward transfers and speed up the download processes.

To this end, we exploit the predictability of large-scale urban vehicular traffic flows, which are known to follow common mobility patterns over a road topology [3], [4], [5]. By studying such traffic dynamics, we determine the way vehicular flows spread over the streets layout and employ this information to guide the APs placement. Next, we first introduce the concept of *crossing volume* that constitutes the basis of our solution, and then employ it to formalize the APs deployment problem as an optimization problem.

A. Crossing volume

Let us imagine that the road topology is represented by a graph where vertices are mapped to intersections and edges to streets connecting them, as in Fig. 2. The graph is undirected, and an edge exists even if the corresponding road is one-way.

Focusing on a particular edge i of the graph, we can track all traffic leaving such edge, in both directions, and draw a map of how vehicular flows (measured in vehicles/s) from i unfold over the road topology. We refer to these flows as the vehicular flows *generated* at i. As an example, in Fig. 2, the red arrows depict flows generated at edge i. Different flows have different size, in vehicles/s as already said, represented by their associated number (values in the example are only illustrative).

Let us now consider a generic edge $k \neq i$, and isolate the flows passing in strict sequence through i and k. In this case, we distinguish the two directions of movement over k, and define two *traversing flows* from i to k:

- $\overrightarrow{f_{ik}}$, the vehicular flow generated at i and subsequently traversing k in the $\rightarrow$ direction;
- $\overleftarrow{f_{ik}}$, the vehicular flow generated at i and subsequently traversing k in the $\leftarrow$ direction.

We do not impose any rule in defining the two directions at k, which, e.g., could be based on vertices numbering or geographical coordinates. Our only concern is that, for each edge, the two directions are unambiguously identified. On the other hand, the direction at i is not specified, and a traversing flow could have visited its generating edge in any direction (including both of them, if flows generated at i in both directions then merge at k in the same direction).

Traversing flows at an edge k can be translated to traversing volumes (measured in vehicles), by evaluating the average time vehicles spend to travel over the entire road segment corresponding to edge k. Also in this case, we can distinguish the two directions of movement, and define the two

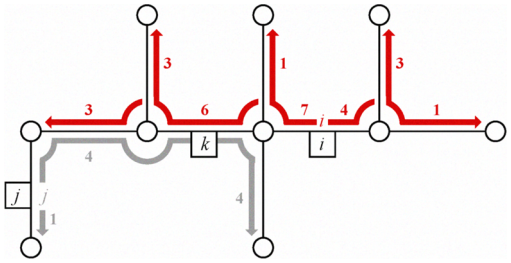


Fig. 2. Sample vehicular flows over a road topology graph. Flows generated at edge i are drawn in red, while those generated at j are in grey. Assuming a travel time $= 1$ at all edges, the partial crossing volume $h^k_{i,j}$ is equal to $min\{6,4\} + min\{0,0\} = 4$, while the crossing volume h_{ij} is $4 + 3 = 7$

travel times $\overrightarrow{t_k}$ and $\overleftarrow{t_k}$, in the $\rightarrow$ and in the $\leftarrow$ directions, respectively. Considering again traversing flows from i, the two corresponding *traversing volumes* are

$$\overrightarrow{v_{ik}} = \overrightarrow{f_{ik}} \cdot \overrightarrow{t_k} \quad , \quad \overleftarrow{v_{ik}} = \overleftarrow{f_{ik}} \cdot \overleftarrow{t_k}$$

and represent the number of vehicles that, on average, are present over k having previously visited i.

Let us now introduce a second group of flows, generated at an edge $j \neq i$, depicted in grey in Fig. 2. The same consideration we made for flows generated at i are valid, and, picked an edge $k \neq j$, we can compute the traversing volumes from j to k, $\overrightarrow{v_{jk}}$ and $\overleftarrow{v_{jk}}$. By considering both sets of flows at once, we can define the *partial crossing volume* of i and j at k, as

$$h^k_{ij} = \begin{cases} min\left\{\overrightarrow{v_{ik}}, \overleftarrow{v_{jk}}\right\} + min\left\{\overleftarrow{v_{ik}}, \overrightarrow{v_{jk}}\right\}, & \text{if } k \neq i, k \neq j \\ 0, & \text{otherwise.} \end{cases}$$

The partial crossing volume h^k_{ij} thus corresponds to the amount of vehicular traffic that merges at edge k, coming from i and j. Note that

- we only account for flows traveling on opposite directions over k, as two cars that happen to be on the same road at the same time in contrary direction always generate (or have generated) a contact, while two cars traveling over the same lane have small probability of meeting, especially in the case of long roads. In other words, opposite flows generate a certain high number of encounters between cars, while flows overlapping in the same direction only result in rare contacts, outcome of strict timings which are hard to predict and quantify. We thus consider the latter contribution as negligible;
- we take the minimum between the two facing traffic volumes, as that is the one imposing a more strict constraint on the number of encounters between vehicles.

Finally, the concept of crossing volume can be unbound from intermediate edges and related to couples of roads only. Assuming that the road topology graph has edges with indices in the set $I = \{1, \ldots, K\}$, we define the *crossing volume* of i and j as

$$h_{ij} = \begin{cases} \sum_{k=1}^{K} h^k_{ij}, & \text{if } i \neq j \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

which implies that $h_{ij} = h_{ji} \geq 0$, $\forall i, j \in I$.

The crossing volume h_{ij} provides a measure of the potential for contact, and thus cooperation, over the entire road network, among vehicles leaving edges i and j.

B. Crossing volume-based APs deployment

We can now exploit the concept of crossing volume to formalize the problem of APs deployment. Let us assume that N APs, with $N < K$, are to be placed over the road topology graph defined before: our problem becomes that of assigning APs to N of the K edges of the graph.

We associate to each edge i a binary decision variable x_i, defined as

$$x_i = \begin{cases} 1, & \text{if an AP is deployed on the road mapped to } i \\ 0, & \text{otherwise,} \end{cases}$$

and refer to their vector as $x = \{x_1, \ldots, x_K\}$.

Since opportunities for cooperation between vehicles are proportional to the crossing volume between each couple of edges, APs should be positioned so to maximize the sum of crossing volumes between each pair of APs over the whole road topology. We can formulate the following mixed-integer quadratic programming (MIQP) problem

$$\max_x \quad f(x) = \frac{1}{2} x' H x \tag{2}$$

$$s.t. \quad x_i \in \{0,1\}, \quad \forall i \in I \tag{3}$$

$$\sum_{i=1}^{K} x_i \leq N \tag{4}$$

$$x_i + x_j \leq 1, \quad \forall j \in \Omega_i,\ j < i,\ \forall i \in I. \tag{5}$$

Here, that in Eq. 2 is the objective function to be maximized, with $H = \{h_{ij}\}$ a $K \times K$ matrix in $\mathbb{R}$, filled with the crossing volumes computed for each couple of edges as in Eq. 1. Also, Eq. 3 states that x_i is a binary variable, $\forall i \in I$, whereas Eq. 4 bounds the overall number of APs to be deployed to N.

The constraint in Eq. 5 avoids that two adjacent edges (i.e., edges with a vertex in common) both host APs. As a matter of fact, Ω_i is the set of indices j labeling edges that are adjacent to edge i: the inequality thus forces the sum of any two decision variables x_i and x_j referring to adjacent edges to be less than or equal to 1. This prevents that APs are placed too close to each other and form locally dense clusters, where only direct communication is used and insufficient space is left in between APs to perform vehicle-to-vehicle transmissions. For the same reason, we only deploy one AP per road.

The problem can be easily extended to the case of incremental APs deployment, in which one or more APs have already been positioned along some roads, and others are to be added. It is sufficient to introduce constraints that force x_i to 1, for all edges i where an AP is already deployed.

Finally, we stress that this formulation solves the APs deployment problem from a large-scale viewpoint, i.e., it allows to determine the roads where APs should be positioned. However, it does not specifies the exact location of each AP over the selected roads. In Sec. V, we will show that such small-scale deployment has in fact a negligible impact on the performance of the system.

III. Carriers selection

Having deployed the APs over the road topology, we can address the carriers selection problem, i.e., answer to the following questions: which of the vehicles in range of an idle AP should be picked as carriers, if any? And which of the downloader cars should such carriers transport data for?

Indeed, the key to the answers is to know in advance whether (and possibly when) one or more cars currently within coverage of the AP will meet a specific downloader vehicle, so to perform the selection that leads to the highest gain in terms of download rate. Also, by choosing carriers depending on their future contacts, the destination of the data becomes constrained to the elected carriers, and the second question above is inherently solved along with the first one.

However, the movement of individual vehicles over urban road topologies cannot be precisely predicted, and we must rely on a probabilistic approach, leveraging again the constant large-scale mobility patterns of vehicular traffic flows. We thus let each AP build a *contacts map*, exploiting historical data about contacts between cars, and use it to estimate the meeting probability between local and downloader vehicles. In the following, we first discuss how contacts map are structured and constructed. Then, we present different carriers selection algorithms and detail how they exploit such contacts maps.

A. Contacts map

We denote as p^k_{Aa} the k-th *production phase* of vehicle a with respect to AP A, i.e., the k-th of the disjoint time intervals during which vehicle a can steadily download data from A [6]. From a specific AP's perspective, we tag production phases as *local* if they involve that particular AP: as an example, p^h_{Bb} is a local production phase for AP B, $\forall\, b, h$.

On the other hand, we label as f^m_{ab} the m-th *forward phase* of vehicle b with respect to vehicle a, i.e., the m-th of the disjoint time intervals during which vehicle b can steadily forward data to vehicle a. We stress that both production and forward phases do not necessarily correspond to actual data transfer, but just to contacts which could be exploited for data transfer.

We also use $t(\cdot)$ to indicate the time at which a production or forward phase starts, and $\Delta t(\cdot)$ to tag its duration. For production phases only, $\alpha(\cdot)$ denotes the historical direction of movement [1] of the vehicle at the beginning of the production phase, while $v(\cdot)$ its speed at that same time. The notation is summarized in Fig. 3.

Structure

A contacts map is a data structure that provides an AP with information on the probability of contact between a vehicle involved in a local production phase and another vehicle. With reference to the example in Fig. 3, the contacts map at AP B

[1]The historical direction is obtained as the angle of movement between the location where the vehicle started its trip, and its current location. This provides a more reliable estimation of the trajectory the vehicle will maintain, with respect to that inferred from the instantaneous direction.

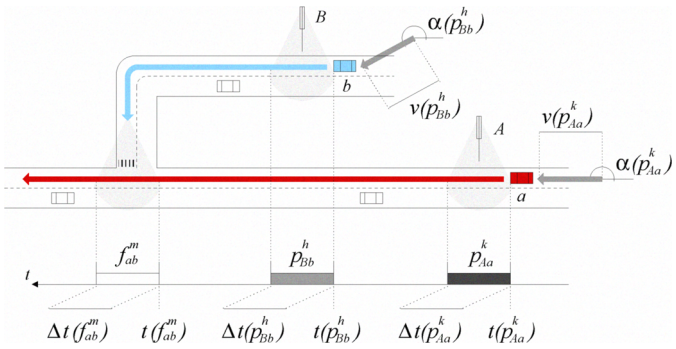


Fig. 3. Notation for contacts map structure and construction

allows B to know the probability of contact between the local vehicle b and the generic vehicle a.

In particular, AP B knows that b started a local production phase p^h_{Bb} at time $t(p^h_{Bb})$, while moving with direction $\alpha(p^h_{Bb})$ and speed $v(p^h_{Bb})$; also, let us assume B has been informed that a started a production phase p^k_{Aa} with AP A at time $t(p^k_{Aa})$, while moving with direction $\alpha(p^k_{Aa})$ and speed $v(p^k_{Aa})$. Then, the contacts map at B allows to associate the couple of production phases (p^h_{Bb}, p^k_{Aa}) to historical data on the encounters between vehicles that had previously generated production phases at the two APs B and A with timings and mobility similar to those of b and a.

More formally, a contacts map is a set of one-to-one associations between *keys*, that encode the significant characteristics of two production phases, and *values*, that store the contacts properties for all couples of production phases that share such characteristics.

The *key* for two generic production phases p^h_{Bb} and p^k_{Aa} is a vector $\mathbb{k}(p^h_{Bb}, p^k_{Aa})$ equal to

$$\left[A, \lfloor\frac{t(p^h_{Bb}) - t(p^k_{Aa})}{\delta t}\rfloor, \lfloor\frac{\alpha(p^h_{Bb})}{\delta\alpha}\rfloor, \lfloor\frac{\alpha(p^k_{Aa})}{\delta\alpha}\rfloor, \lfloor\frac{v(p^h_{Bb}) - v(p^k_{Aa})}{\delta v}\rfloor\right],$$

where $\delta\alpha$, δt and δv are the units (in degrees, seconds, and meters/second, respectively) used to discretize angles, time and speed. A couple of production phases is thus characterized by the identity of the AP involved in the second production phase, the time elapsed between the start of the two production phases, the direction of the two vehicles at the beginning of the respective production phases, and the difference between their speeds at that same time. We stress that the identity of the other AP is not necessary, as the first production phase is always a local one.

A *value* is instead a vector of four fields:

1) n_{opps}, the number of contact opportunities, i.e., the number of times that the AP observed a couple of production phases with characteristics as from the associated key;
2) n_{cons}, the number of actual contacts, i.e., the number of times that vehicles from the aforementioned couples of production phases actually generated a forward phase;
3) t_{del}, the average time elapsed between the start of the last production phase and the start of the forward phase, when the latter has indeed occurred;
4) t_{dur}, the average duration of the forward phase, when it has indeed occurred.

It is to be noticed that each AP builds its own contacts map, in which it stores only values associated to keys where the first production phase, as already said, is a local one. As an example, an AP B will only store values for keys of the type $\mathbb{k}(p^h_{Bb}, p^k_{Aa})$, $\forall h, b, k, A, a$. The rationale is that local production phases are the only vehicle-to-infrastructure contacts that an AP can exploit for carriers selection, and are thus the only ones an AP is interested in recording data for.

Construction

The steps for the construction of the contacts map at an AP are best described by means of an example, so we consider once more the situation depicted in Fig. 3.

When production phase p^k_{Aa} starts, A logs the time $t(p^k_{Aa})$, the relative vehicle identifier a, its historical direction $\alpha(p^k_{Aa})$, and its current speed $v(p^k_{Aa})$. This information is shared with other APs in the same area [2], and updated, when the production phase ends, with the information on the duration $\Delta t(p^k_{Aa})$. This way, when the production phase p^k_{Aa} terminates, AP B has memory of the event, including all related details.

Similarly, at the beginning of p^h_{Bb}, B records and shares with other APs identical information on such production phase, which is then updated at the end of p^h_{Bb}.

At that time, as every other AP does at the end of its own local production phases, AP B checks whether p^h_{Bb} can be considered as an opportunity for cooperative download with respect to other production phases it is aware of. It is so if another production phase is found to be compatible with p^h_{Bb} (see next for a definition of production phases compatibility). In our case, let us assume that the production phase p^k_{Aa} is compatible with p^h_{Bb}, then B looks in its local contacts map for the value associated to key $\mathbb{k}(p^h_{Bb}, p^k_{Aa})$. If the entry is not found, it is created, and, in both cases, its n_{opps} field is incremented.

Proceeding in our example, vehicle b later on meets vehicle a, generating the forward phases f^m_{ab} and f^m_{ba}. We focus on the first one, as it is that of interest in our example. Both vehicles record the forward phase start time $t(f^m_{ab})$, as well as the other vehicle's identifier. Upon loss of contact, a and b also log the forward phase duration $\Delta t(f^m_{ab})$. These same cars upload these information, together with similar data on all other forward phases they have experienced, to the next AP they encounter, which will again share them with APs in the same area.

When AP B is notified of the forward phase f^m_{ab}, it tries to understand if f^m_{ab} can be related to any of the opportunities it has previously recorded. Thus, B scans its database for local production phases compatible with f^m_{ab} (see next for a definition of production phase compatibility with a forward phase). Assuming that p^h_{Bb} satisfies the compatibility constraint, B then looks for production phases of vehicle a, with

[2]For the purpose of cooperative download, we do not need all APs to know all vehicle-to-infrastructure contacts. Instead, it is sufficient that APs are informed of contacts occurring in a limited area around them, which guarantees system scalability. In our tests, we considered this limited area to have a radius of a few kilometers, matching the size of the simulated scenarios.

any AP, that are compatible with p^h_{Bb}. B finds again p^k_{Aa}, and thus finally relates f^m_{ab} to the couple of production phases (p^h_{Bb}, p^k_{Aa}). At this point, B retrieves the value associated to the key $\Bbbk(p^h_{Bb}, p^k_{Aa})$, and updates the n_{cons}, t_{del} and t_{dur} fields. The first is incremented by one, to record that the opportunity previously stored actually generated a forward phase. The second and the third elements are updated using samples $t(f^m_{ab}) - t(p^h_{Bb})$ and $\Delta t(f^m_{ab})$, respectively. The way these last updates are performed depends on the desired detail level of information on the vehicle-to-vehicle contact: in our case, we opted for keeping track of the mean of the samples.

Phases compatibility

Phases compatibility rules determine when two production phases generate an opportunity for cooperative download, as well as when a forward phase can represent a contact for a local production phase.

These rules formally relate phases in a way that avoids inconsistencies in the resulting contacts maps, an event otherwise common, especially when considering that phases often overlap in time and/or refer to a same AP (i.e., it can be that A and B in all previous discussions are indeed the same AP).

We first introduce the set of rules for the compatibility of a local production phase with respect to a forward phase. A local production phase p^h_{Bb} is said to be compatible with a forward phase f^m_{ab} if the following conditions are verified:

1) the forward phase has ended after the end of the local production phase
$$t(f^m_{ab}) + \Delta t(f^m_{ab}) > t(p^h_{Bb}) + \Delta t(p^h_{Bb}),$$
as b must receive the data from B before it can forward them to a. Note that the indices already imply that, for the phases to be compatible, the same vehicle b must realize the production phase with B and be the potential carrier in the forward phase;
2) the forward phase is the first involving a and b and satisfying rule 1 to have terminated after the end of the local production phase
$$\nexists n \mid t(f^n_{ab}) + \Delta t(f^n_{ab}) > t(p^h_{Bb}) + \Delta t(p^h_{Bb}), \quad t(f^n_{ab}) < t(f^m_{ab}),$$
which guarantees that at most one forward phase is associated to each production phase;
3) the local production phase at B is the last involving b and satisfying rule 1 that started before the forward phase's end
$$\nexists n \mid t(f^m_{ab}) + \Delta t(f^m_{ab}) > t(p^n_{Bb}) + \Delta t(p^n_{Bb}), \quad t(p^n_{Bb}) > t(p^h_{Bb}),$$
which guarantees that at most one production phase is associated to each forward phase.

Then, we introduce the rules that define the compatibility between two production phases. A production phase p^k_{Aa} is said to be compatible with a local production phase p^h_{Bb} if the following conditions are verified:

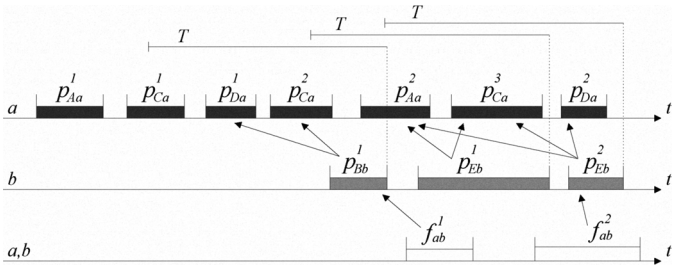


Fig. 4. Example of phases compatibility. Arrows show production phases of car a that are compatible with local production phases of car b, as well as which of the latter are compatible with forward phases of b to a

1) the first production phase has ended before the end of the local production phase
$$t(p^h_{Bb}) + \Delta t(p^h_{Bb}) > t(p^k_{Aa}) + \Delta t(p^k_{Aa}),$$
which accounts for the fact that an AP can only destine carry&forward data to production phases it is aware of;
2) the first production phase has ended at most a time T before the end of the local production phase
$$t(p^h_{Bb}) + \Delta t(p^h_{Bb}) - t(p^k_{Aa}) - \Delta t(p^k_{Aa}) \leq T,$$
that avoids considering too old production phases;
3) the first production phase is the last involving a and A satisfying rules 1 and 2 to have ended before the end of the local production phase
$$\nexists n \mid 0 < t(p^h_{Bb}) + \Delta t(p^h_{Bb}) - t(p^n_{Aa}) - \Delta t(p^n_{Aa}) \leq T, \quad t(p^n_{Aa}) > t(p^k_{Aa}),$$
which guarantees that at most one production phase involving same vehicle/AP couple is associated to each local production phase.

Examples of phases compatibility resulting from the rules listed above are provided in Fig. 4.

B. Carriers selection algorithms

Once built, a contacts map can be exploited by an AP to select local vehicles as data carriers for cooperative download, by retrieving their contact probability estimates with respect to downloader cars.

Firstly, it is necessary that APs know which cars in their surroundings are interested in some content. Thus, every time a downloader vehicle starts a production phase, the fact that it is requesting data is attached to the usual information on the production phase that the local AP shares with other APs. This way, each AP can track downloader vehicles through their production phases history.

Thanks to such knowledge, an AP that has active local production phases can compute the *delivery potential* $\mathbb{p}_a$ resulting from electing one (or some, or all) of the local vehicles as carrier(s) for data destined to a specific downloader vehicle a. The delivery potential is obtained as the sum of the individual contact probabilities $\mathbb{p}_b$, derived from assigning data for the downloader a to each elected local carrier b [3].

[3]Leveraging the broadcast nature of the wireless channel, a single multicast transmission is sufficient to transfer the same data to all elected local carriers.

```
01 set p equal to P_min
02 for each downloader vehicle a do
03   if a is in range of B do
04     if a is closer to B than previous direct downloaders do
05       select a as destination for direct transfer
06       select no vehicles as carriers for carry&forward transfer
07       set p equal to ∞
08     done
09   else
10     for each production phase p^k_Aa of a do
11       until all on-going local production phases are not processed do
12         update delivery potential p^k_Aa
13         update carriers list c̄^k_Aa
14       done
15       if p^k_Aa is the highest potential computed for a do
16         set p_a equal to p^k_Aa
17         set c̄_a equal to c̄^k_Aa
18       done
19     done
20     if p_a is strictly higher than p do
21       select a as destination for carry&forward transfer
22       select vehicles in c̄_a as carriers for carry&forward transfer
23       set p equal to p_a
24     done
25   done
26 done
```

Fig. 5. Pseudocode for carriers selection at AP B

```
01 get next on-going local production phase p^h_Bb
02 set p_b equal to a random value ∈ (0, 1]
03 add p_b to delivery potential p^k_Aa
04 add b to carriers list c̄^k_Aa
05 mark local production phase p^h_Bb as processed
```

Fig. 6. Blind pseudocode for $\mathbb{p}^k_{Aa}$, $\bar{\mathbb{c}}^k_{Aa}$ update

The process is repeated for each known downloader car, and, in the end, the downloader vehicle associated with the highest delivery potential $\mathbb{p}$ [4] is chosen for carry&forward transfer through local carriers that contributed to $\mathbb{p}$.

The framework for carriers selection run at a generic AP B is shown as pseudocode in Fig. 5. There, priority is always given to direct data transfers to downloader cars, and fairness among them is provided by always picking the vehicle that is closest to the AP. The parameter $\mathbb{P}_{min}$ controls the minimum delivery potential required to attempt cooperative download through local carriers. The value of such parameter (line 01 in Fig. 5), together with the way the delivery potential $\mathbb{p}^k_{Aa}$ associated to the downloader production phase p^k_{Aa} and its relative carriers list $\bar{\mathbb{c}}^k_{Aa}$ are updated (lines 12 and 13 in Fig. 5), distinguish the following carriers selection algorithms.

The **Blind** carriers selection algorithm aims at fully exploiting the airtime available at APs, by delivering data to all available local carriers whenever possible. This algorithm does not make use of the contacts map, but randomly chooses a downloader car as the destination of the data: we thus employ it as a benchmark for the other schemes. The pseudocode for potential and carriers list updating is outlined in Fig. 6, while $\mathbb{P}_{min}$ is set to 0, so that cooperative download is attempted even in presence of a single local carrier.

The **p-Driven** carriers selection algorithm is a probability-driven version of the Blind algorithm above. It again tries to exploit as much as possible the APs wireless resources, but this time cooperative download destinations are selected according to the delivery potential obtained from the contacts map.

[4]Note that $\mathbb{p}$ can be > 1, to counter uncertainties in probability estimates.

```
01 get next on-going local production phase p^h_Bb
02 get key k(p^h_Bb, p^k_Aa)
03 if a contacts map entry for such key exists do
04   get relative value {n_opps, n_cons, t_del, t_dur}
05   set p_b equal to n_cons / n_opps
06   add p_b to delivery potential p^k_Aa
07   add b to carriers list c̄^k_Aa
08 done
09 mark local production phase p^h_Bb as processed
```

Fig. 7. p-Driven pseudocode for $\mathbb{p}^k_{Aa}$, $\bar{\mathbb{c}}^k_{Aa}$ update

```
01 get next on-going local production phase p^h_Bb
02 get key k(p^h_Bb, p^k_Aa)
03 if a contacts map entry for such key exists do
04   get relative value {n_opps, n_cons, t_del, t_dur}
05   set p_b equal to n_cons / n_opps
06   if p_b is equal to or greater than P_ind do
07     add p_b to delivery potential p^k_Aa
08     add b to carriers list c̄^k_Aa
09   done
10 done
11 mark local production phase p^h_Bb as processed
```

Fig. 8. p-Constrained pseudocode for $\mathbb{p}^k_{Aa}$, $\bar{\mathbb{c}}^k_{Aa}$ update

As a matter of fact, carry&forward data is consigned by each AP to all available local vehicles, and destined to the downloader vehicle which maximizes the sum of its contact probabilities with all the local carriers, as detailed in the pseudocode of Fig. 7. We stress that non-compatible production phases generate keys that are not present in the contacts map, and are thus not considered for cooperative download.

As the p-Driven algorithm is designed to exploit carry&forward whenever there is a minimal chance of delivery, $\mathbb{P}_{min}$ is set to 0: this allows cooperative download even in presence of very small delivery potentials. We expect the p-Driven scheme to be more precise in carriers selection than the Blind one, but not in absolute terms.

The **p-Constrained** carriers selection algorithm builds on top of the p-Driven scheme, adding constraints on probabilities, as from the pseudocode in Fig. 8. In particular, local vehicles with individual contact probability $\mathbb{p}_b$ lower than $\mathbb{P}_{ind} > 0$ are not considered for data carrying, and $\mathbb{P}_{min}$ is set to a value higher than 0, so that downloader vehicles with delivery potential $\mathbb{p}_a$ lower than $\mathbb{P}_{min}$ are discarded.

Thanks to the lower bounds on individual probability and delivery potential, the p-Constrained algorithm is expected to further increase the delivery precision and reduce the load at APs with respect to the p-Driven scheme. However, quality could come at cost of quantity, as the thresholds may hinder potentially successful cooperation among vehicles.

The **(p,t)-Constrained** carriers selection algorithm adds time constraints to the probability constraints of the p-Constrained scheme. It introduces a distributed database $\bar{\mathbb{t}}_a$, maintained for each active downloader vehicle a by APs in a same area, controlling what portions of a's time are assigned to which specific carriers [5]. As shown in the pseudocode in Fig. 9, the (p,t)-Constrained algorithm processes local

[5]We recognize that maintaining such database can pose synchronization and consistency issues, whose management is out of the scope of this paper. We however stress that we do not require frequent updates or high consistency in $\bar{\mathbb{t}}_a$, since the update periodicity is in the order of seconds and errors in the database are overshadowed by inaccuracy in the contact estimation.

```
01 set $\mathbb{p}_{max}$ equal to $\mathbb{P}_{ind}$
02 for each on-going local production phase $p^h_{Bb}$ do
03   if $p^h_{Bb}$ is marked as processed do
04     continue
05   done
06   get key $\mathbb{k}(p^h_{Bb}, p^k_{Aa})$
07   if a contacts map entry for such key exists do
08     get relative value $\{n_{opps}, n_{cons}, t_{del}, t_{dur}\}$
09     set $\mathbb{p}_b$ equal to $\frac{n_{cons}}{n_{opps}}$
10     if $\mathbb{p}_b$ is equal to or greater than $\mathbb{p}_{max}$ do
11       set $p$ equal to production phase $p^h_{Bb}$
12       set $v$ equal to production phase $p^h_{Bb}$ vehicle $b$
13       set $\mathbb{p}_{max}$ equal to $\mathbb{p}_b$
14     done
15   done
16 done
17 if $\mathbb{p}_{max}$ is equal to $\mathbb{P}_{ind}$ do
18   mark all unmarked local production phases as processed
19 else
20   get key $\mathbb{k}(p, p^k_{Aa})$
21   get relative value $\{n_{opps}, n_{cons}, t_{del}, t_{dur}\}$
22   for each time step $t \in \left[\lfloor\frac{t(p)+t_{del}}{\mathbb{T}}\rfloor, \lfloor\frac{t(p)+t_{del}+t_{dur}}{\mathbb{T}}\rfloor\right]$ do
23     if $\mathbb{p}_{max}$ is lower than or equal to $\mathbb{P}_{min} - \bar{\mathbb{t}}_a(t)$ do
24       add $\mathbb{p}_{max}$ to delivery potential $\mathbb{p}^k_{Aa}$
25       add $v$ to carriers list $\bar{\mathbb{c}}^k_{Aa}$
26       set $\bar{\mathbb{t}}_a(t)$ equal to $min\{\bar{\mathbb{t}}_a(t) + \mathbb{p}_{max}, \mathbb{P}_{min}\}$
27       continue
28     done
29   done
30   mark local production phase $p$ as processed
31   if $\mathbb{p}^k_{Aa}$ is equal to or greater than $\mathbb{P}_{min}$ do
32     mark all unmarked local production phases as processed
33   done
34 done
```

Fig. 9. (p,t)-Constrained pseudocode for $\mathbb{p}^k_{Aa}$, $\bar{\mathbb{c}}^k_{Aa}$ update

vehicles b in decreasing order of contact probability with the downloader car a, skipping those with probability lower than $\mathbb{P}_{ind}$ (lines 02 to 16 in Fig. 9). Every time the unprocessed local vehicle with maximum contact probability $\mathbb{p}_{max}$ is processed, the algorithm exploits information on the average time to contact (t_{del}) and contact duration (t_{dur}) to predict the time interval during which the local vehicle will meet the downloader car a. Then, it discretizes time with step $\mathbb{T}$, and tries to fit the estimated contact probability $\mathbb{p}_{max}$ in one of the time steps within the aforementioned time interval (lines 20 to 30 in Fig. 9).

The process is terminated when either the minimum required delivery potential $\mathbb{P}_{max}$ has been reached (lines 31 to 33 in Fig. 9), or no more local vehicles are available (lines 17 to 19 in Fig. 9). In the second case, the delivery potential constraint is not fulfilled, as $\mathbb{P}_{min}$ is higher than zero, and thus no carriers can be selected for the current production phase p^k_{Aa} (see line 20 in Fig. 5).

The (p,t)-Constrained algorithm therefore employs information about contact times to improve the delivery precision, reducing the data carriers involved in the cooperative download.

IV. Chunk scheduling

Upon selection of a carry&forward destination and its associated local carriers, an AP must decide on which portion of the data the downloader is interested in are to be transfered to the carriers.

We assume that each content is divided into *chunks*, i.e., small portions of data that can be transfered as a single block from the AP to the carriers, and then from the latter to the destination. We introduce three chunk scheduling schemes that embody increasing levels of redundancy, presented next.

Global chunk scheduling assumes that APs maintain per-vehicle distributed chunk databases, similar to the time databases $\bar{\mathbb{t}}_a$ introduced before. These databases store information on which chunks have already been scheduled for either direct or carry&forward delivery to each downloader vehicle.

The Global scheme then completely distributes the chunk scheduling among APs, since it forces an AP to pick a new, unscheduled chunk every time it performs a direct or carry&forward transfer. In other words, each chunk is scheduled for transfer just once in the entire network. We stress that, even then, multiple carriers can be given the same chunk, as carriers selection algorithms can (and usually do) identify more than one vehicle for a single carry&forward transfer.

Hybrid chunk scheduling does not require the aforementioned per-vehicle chunk database, as it forces non-overlapping scheduling (i.e., it always selects a new, unscheduled chunk) only in case of direct transfers. Since direct transfers occur during contacts between APs and the downloader vehicle, the chunk scheduling history can be easily maintained at vehicles, and communicated to the current AP at the beginning of the local production phase. On the other hand, carry&forward transfers can overlap. In fact, in case of a data transfer to carriers, an AP picks the first chunk not already received by the selected downloader car and that *it* has not yet scheduled, ignoring the carry&forward scheduling at other APs. The Hybrid scheme is implicitly more redundant that the Global one, as different APs independently assign carriers for a same chunk.

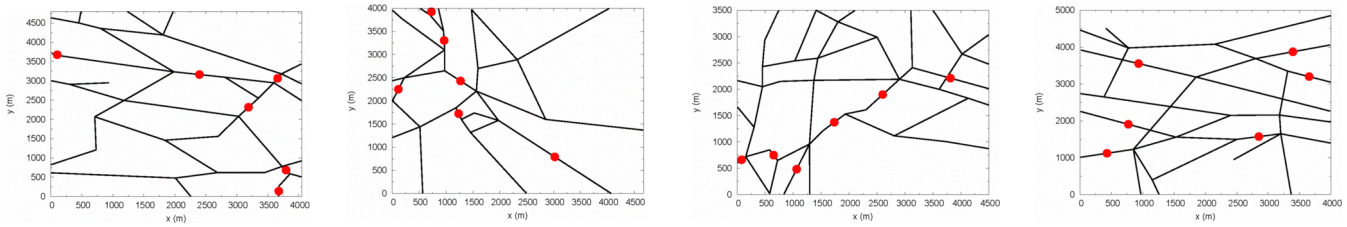
Local chunk scheduling is similar to the Hybrid scheme, but it also allows overlapping between direct and carry&forward transfers. This means that an AP can employ a direct transfer to a downloader car to fill gaps in its scheduled chunk list, or, in other words, the AP can transfer chunks already scheduled for carry&forward delivery, but not yet received. Local scheduling is thus the most redundant among the schemes we propose, trading some cooperative download potential for increased reliability in data delivery.

V. Performance evaluation

We conducted an extensive simulative evaluation of the cooperative download framework outlined in the previous sections, considering large-scale scenarios with realistic vehicular mobility over real-world road topologies. In the following, we first present the simulation scenarios and network settings, and then discuss the results of the performance analysis.

A. Scenarios and settings

To assess the performance of the cooperative download solutions presented in the previous sections, we selected real-world road topologies from the area of Zurich, Switzerland. This choice was mainly driven by the availability of large-scale microscopic-level traces of the vehicular mobility in such region, generated at the Computer Science Department of ETH Zurich [7].



(a) West Zurich (b) Central Zurich (c) North Zurich (d) Schlieren

Fig. 10. The four road topology scenarios considered in our study, representing urban, suburban, and rural areas in the canton of Zurich, Switzerland

The simulation techniques and mobility models employed to generate the traces allow to reproduce vehicular movement over very large road topologies, yet with a good degree of precision. The traces replicate macroscopic patterns of real-world vehicular traffic flows, made up of thousand of cars, as well as microscopic behaviors of individual drivers in urban environments, such as pauses at intersections that depend roads capacity and congestion. This macro- and micro-mobility realism is important in our study, since, on the one hand, we exploit large-scale properties of urban vehicular mobility in designing the cooperative download system, and, on the other, realistic small-scale mobility is required to reproduce vehicle-to-vehicle and vehicle-to-AP networking interactions.

We focused on four scenarios, representing urban, suburban, and rural areas within and nearby the city of Zurich. All the areas considered cover surfaces between 15 and 20 km^2, and frame several tens of kilometers of major roads. The road topology graphs are shown in Fig. 10, where we also marked with red dots the edges selected for APs deployment by the crossing volume-based strategy described in Sec. II, for the case when $N = 6$.

The elevate number of vehicles, reaching several thousands of units in some of the scenarios, coupled with the complexity of the framework, prevented us from using a traditional network simulator, such as *ns-2*. Instead, we developed a dedicated simulator, which employs the realistic mobility extracted from the Zurich traces, models a random access channel contention, and implements all the schemes described before, but replaces the packet-level simulation with a more scalable chunk-level one and avoids reproducing the detailed behavior of the entire network stack.

The system parameters were set for all simulations to $\delta t = 5s$, $\delta\alpha = 45°$, $\delta v = 5\frac{m}{s}$, $T = 500s$, $\mathbb{P}_{min} = 2.5$, $\mathbb{P}_{ind} = 0.5$, $\mathbb{T} = 1s$, while ten downloader cars are on average present at the same time over the road topology. Also, in the following and where not stated otherwise, we assume a crossing volume-based APs deployment, a number of APs $N = 6$, a (p,t)-Constrained carriers selection, and a Global chunk scheduling as the default simulation settings.

For each simulation we performed one run to train the framework (i.e., gather data for AP deployment and build contacts maps at APs), and ten runs to collect statistics. Each run was around three hours long in simulation time, featuring various vehicular density conditions over such a time span. For all results we measured 99% confidence intervals, reported as error bars in the plots.

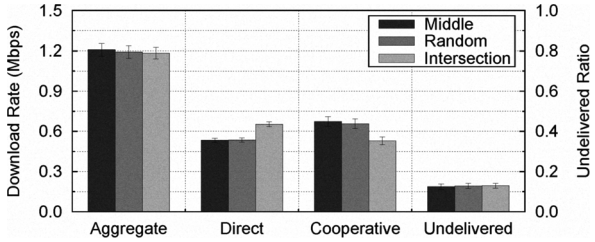


Fig. 11. Download rates and undelivered chunks ratios for different road-level deployments, averaged over all scenarios

The metrics we are interested in evaluating are:

- the *aggregate download rate*, i.e., the average file transfer speed experienced by downloader vehicles traveling through the scenario. Such rate is the sum of a *direct* rate, due to data downloaded from APs, and a *cooperative* rate, due to carry&forward transfers. We underline that, during both production and forward phases, we assumed a net data transmission rate over the wireless channel equal to 5 Mbps for a transmission range of 100 m, values consistent with the outcome of real-world experiments [8]. The maximum download rate achievable by a vehicular user is thus 5 Mbps and corresponds to the case of a car continuously receiving data during its whole trip through the simulation scenario;
- the *undelivered chunk ratio*, i.e., the average ratio of chunks that are not delivered to a downloader vehicle, computed over all those scheduled for that vehicle.

B. Results

As a very first step, we assess the impact of small-scale deployment of APs, by testing different road-level APs placements under the default settings outlined before: the roads where APs are placed are thus those selected by the crossing volume technique, but the exact position of the APs over them varies according to three reference policies. More precisely, the *middle* strategy places the AP equidistant from the intersections that end the road segment, the *intersection* strategy deploys the AP at the most crowded of such intersections, while the *random* strategy picks a random location over the road segment. In fact, Fig. 11 shows that the relevance of road-level deployment is minimal, as the three schemes achieve almost identical download rate and undelivered chunk ratio. The only notable difference is in that the intersection strategy favors direct downloads and penalizes cooperative ones: indeed, crossroads are characterized by high densities of slow vehicles [9], and placing APs there favors AP-to-vehicle

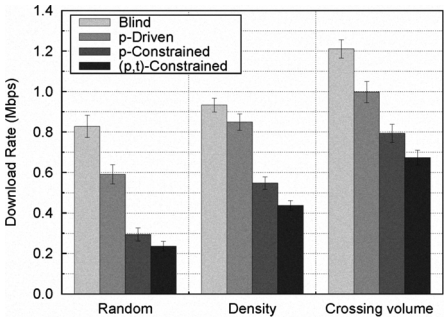


Fig. 12. Cooperative download rates for different APs deployment strategies and carriers selection schemes, averaged over all scenarios

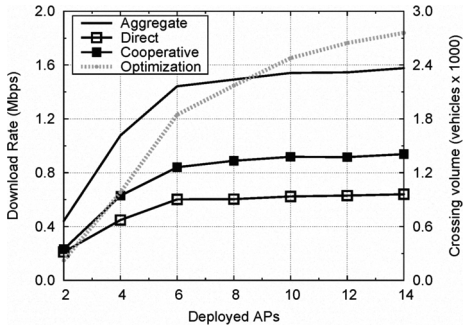


Fig. 13. Optimization result and download rates, versus the number of deployed APs in the West Zurich scenario

transfers. At the same time, however, it deprives vehicles of transfer opportunities, since intersections also represent network clustering points where car-to-car contacts occur frequently [9], thus reducing the cooperative download rate. In the remainder of the paper, we will consider APs to be deployed at the intermediate point of road segments, as this appears to bring an advantage over the other solutions, if minimal, in terms of aggregate rate.

Secondly, we evaluate the performance of the crossing volume-based APs deployment, by introducing two benchmark strategies: a *random* APs positioning scheme, that casually places APs over the road topology, and a *density-based* APs deployment technique, that distributes APs at the most traffic-congested crossroads. The crossing volume-based APs deployment strategy proves a clear winner over the two benchmark techniques, in Fig. 12: no matter the carriers selection scheme considered, the crossing volume-based strategy achieves a cooperative download rate that is consistently higher than that recorded with the other APs deployments, demonstrating that knowledge of traffic flows can be exploited to foster the potential for cooperation among vehicles.

As a final test on the APs deployment, we vary the number of APs over the road topology, and observe how this impacts on the average download rate experienced by vehicular users. In Fig. 13, we can see that both the direct and cooperative download rates grow with the number of APs, faster at first, and then slower and slower. Indeed, once enough APs have been deployed, all the major vehicular flows are covered and the potential for cooperation is almost fully exploited: additional APs end up in minor roads, where they scarcely contribute to both direct and cooperative data transfers. This

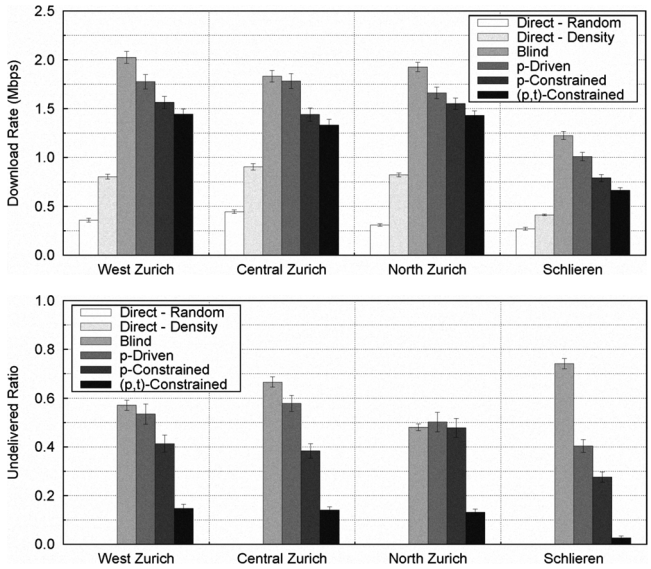


Fig. 14. Aggregate download rates (top) and undelivered chunks ratios (bottom) for different carriers selection schemes

is consistent with the result of the crossing volume-based optimization problem, which draws a curve similarly steep at first and leveling afterwards.

Fig. 14 provides a more comprehensive evaluation of the cooperative download framework we defined, whose performance, under different carriers scheduling strategies, are benchmarked against those of direct-download-only systems where APs are deployed according to both the random and the density-based policies introduced above.

The first observation is that cooperative download works. In the top plot of Fig. 14, we can observe that opportunistic collaborations among vehicles increase the average download rate achieved by means of AP-to-vehicle transfers of around 100% in the worst case (i.e., comparing the cooperative (p,t)-Constrained scheme and the direct download under a density-based APs deployment), but the gain can in fact reach up to several times such value.

The comparison among different carriers selection solutions shows instead that the lower is the targeted delivery precision, the higher the resulting average download rate. However, less accurate schemes tend to pay a much greater cost in terms of scheduled but undelivered chunks, depicted in the lower plot of Fig. 14, than the advantage they introduce in terms of download rate. With the Blind carriers selection algorithm, approximately two chunks are never delivered for each that reaches its destination. On the contrary, the (p,t)-Constrained scheme is able to deliver 85% circa of the scheduled chunks.

If one were only interested in the raw download rate, without any regard for network resources, a blind selection would be the best choice. However, efficiency in resources utilization is often mandatory, and a moderate reduction in the download rate can be easily traded for much higher delivery precision and lower network load. In any case, the diverse carriers selection schemes provide a whole range of intermediate solutions to select among.

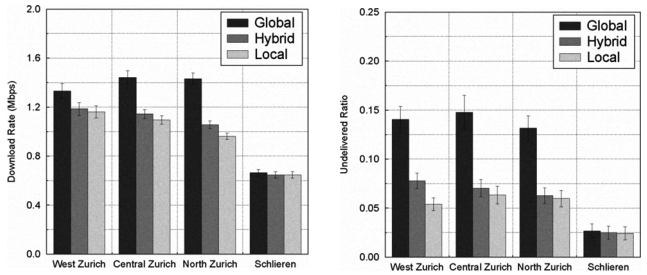


Fig. 15. Aggregate download rates (left) and undelivered chunks ratios (right) for different chunk scheduling schemes

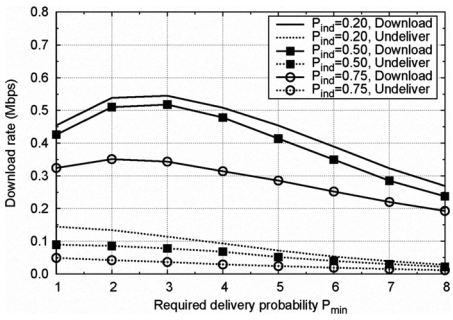


Fig. 16. Cooperative download rates for different configurations of the $\mathbb{P}_{min}$ and $\mathbb{P}_{ind}$ parameters in the (p,t)-Constrained algorithm

In fact, an even finer resolution is allowed by chunk scheduling. The results in Fig. 14 refer to the case in which we employ a Global chunks scheduling, that, as already discussed, is the less redundant among the algorithms introduced in Sec. IV. Therefore, as portrayed in Fig. 15, the failed delivery rate can be noticeably reduced by replacing it with a Hybrid or Local scheduling. In particular, a Local scheduling is shown to reduce by more than a half the number of undelivered chunks, in front of a 10-20% reduction in the download rate, under the (p,t)-Constrained carriers selection technique. This configuration thus provides a x10 reduction in the ratio of undelivered chunks, with respect to a Blind carriers selection with Global chunk scheduling.

As a concluding remark, we briefly discuss the calibration of carriers scheduling algorithms. The Blind and p-Driven schemes do not need any configuration, while both p-Constrained and (p,t)-Constrained rely on the required delivery potential $\mathbb{P}_{min}$ and the individual contact probability $\mathbb{P}_{ind}$. Fig. 16 makes it clear how some redundancy is instrumental to combat uncertainty in the contact probability, as a $\mathbb{P}_{min}$ higher than one increases the cooperative download rate. However, a too high $\mathbb{P}_{min}$ risks to hinder favorable cooperation opportunities with small gain in terms of delivery precision. Similarly, a $\mathbb{P}_{ind}$ higher than zero excludes unreliable carriers from the process, but a $\mathbb{P}_{ind}$ too near to one discards too many local vehicles. The values we employed in the tests, $\mathbb{P}_{min} = 2.5$ and $\mathbb{P}_{ind} = 0.5$, are consistent with these considerations.

VI. Related work

Cooperative download in vehicular network has been first studied by Nandan *et al.* [10], who introduced SPAWN, a protocol for cooperative content retrieval and sharing among users aboard vehicles. However, their approach is hardly comparable with ours, as it (i) considers unidirectional traffic over highways, while we focus on more challenging urban environments, and (ii) proposes a solution inspired by peer-to-peer networking, assuming that all on-road vehicles are active downloaders of the same contents, whereas our work is more oriented to opportunistic communications. The highway scenario is also the only considered by Trullols *et al.* [11], while Zhao *et al.* [12] examine an urban environment, but target small-sized upload transfers from vehicles to static gateways. Techniques for Medium Access Control [13] and network coding [14] in vehicular cooperative download could complement the solutions we outlined in this paper.

As far as APs deployment is concerned, Marfia *et al.* [15] studied the impact on routing of random APs deployments in urban road topologies. We proved that such an approach is inefficient when targeting cooperative download. Solutions for APs deployment over road topologies have been proposed by Ding *et al.* [16], to favor delay-tolerant data exchange among vehicles, and Lochert *et al.* [17], for information dissemination purposes. The diverse goals in these works lead to in different approaches and results with respect to those we presented.

VII. Conclusions

We presented a complete study of cooperative download in urban vehicular environments, proposing solutions for APs deployment, carriers selection and chunk scheduling. We proved the feasibility of cooperative download and demonstrated the significant performance improvements it can bring to users.

References

[1] F. Aidouni, M. Latapy, C. Magnien, *"Ten weeks in the life of an eDonkey server"*, Hot-P2P'09, Rome, Italy, May 2009.
[2] K. Fall, *"A delay-tolerant network architecture for challenged Internets"*, ACM Sigcomm'03, Karlsruhe, Germany, August 2003.
[3] A. Fleisher, *"On prediction and urban traffic"*, Papers in Regional Science, vol.7, no. 1, December 1961.
[4] K. Ashok, M.E. Ben-Akiva, *"Estimation and prediction of time-dependent origin-destination flows with a stochastic mapping of path flows and link flows"*, Transportation Science, vol. 36, no. 2, May 2002.
[5] H. Yin, S.C. Wong, J. Xu, C.K. Wong, *"Urban traffic flow prediction using a fuzzy-neural approach"*, Transportation Research C, vol. 10, no. 2, April 2002.
[6] J. Ott, D. Kutscher, *"Drive-thru Internet: IEEE 802.11b for automobile users"*, IEEE Infocom'04, Hong Kong, China, March 2004.
[7] N. Cetin, A. Burri, K. Nagel, *"A large-scale multi-agent traffic microsimulation based on queue model"*, STRC'03, Ascona, Switzerland, March 2003.
[8] R. Gass, J. Scott, C. Diot, *"Measurements of in-motion 802.11 networking"*, IEEE WMCSA/HotMobile'06, Washington, USA, April 2006.
[9] M. Fiore, J. Härri, *"The networking shape of vehicular mobility"*, ACM MobiHoc'08, Hong Kong, China, May 2008.
[10] A. Nandan, S. Das, G. Pau, M. Gerla, M.Y. Sanadidi, *"Co-operative downloading in vehicular ad-hoc wireless networks"*, WONS'05, St.Moritz, Switzerland, January 2005.
[11] O. Trullols-Cruces, J. Morillo, J. Barcelo-Ordinas, J. Garcia-Vidal, *"A Cooperative Vehicular Network Framework"*, IEEE ICC'09, Dresden, Germany, June 2009.
[12] J. Zhao, G. Cao, *"VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks"*, IEEE INFOCOM'06, Barcelona, Spain, April 2006
[13] J. Zhang, Q. Zhang, W. Jia, *"A novel MAC protocol for cooperative downloading in vehicular networks"*, IEEE GLOBECOM'07, Washington, USA, December 2007.
[14] S. Ahmed, S.S. Kanhere, *"VANETCODE: network coding to enhance cooperative downloading in vehicular ad hoc networks"*, ACM IWCMC'06, Vancouver, Canada, July 2006.
[15] G. Marfia, G. Pau, E. Giordano, E. De Sena, M. Gerla, *"Evaluating vehicle network strategies for downtown Portland: opportunistic infrastructure and importance of realistic mobility models"*, ACM MoBiOpp'07, San Juan, Puerto Rico, June 2007.
[16] Y. Ding, C. Wang, L. Xiao, *"A Static-Node Assisted Adaptive Routing Protocol in Vehicular Networks"*, ACM VANET'07, Montreal, Canada, September 2007.
[17] C. Lochert, B. Scheuermann, C. Wewetzer, A. Luebke, M. Mauve, *"Data aggregation and roadside unit placement for a vanet traffic information system"*, ACM VANET'08, S.Francisco, USA, September 2008.