Prefetching of mobile devices information - a DNS perspective

Antoine Bernard*†, Michel Marot†, Monique Becker†, Mohammed Laroui†‡
Sandoche Balakrichenan*, Hossam Afifi†, Hassine Moungla†‡, Benoit Ampeau*
*Afnic - Email : {sandoche.balakrichenan,benoit.ampeau}@afnic.fr
†Samovar, Télécom SudParis, Institut Polytechnique de Paris, 91120 Palaiseau, France
Email: {antoine_bernard,mohammed.laroui,michel.marot,hassine.moungla,hossam.afifi,monique.becker}@telecom-sudparis.eu
‡LIPADE, Université Paris Descartes, Université Sorbonne Paris Cité, Paris, France
Email: {mohammed.laroui,hassine.moungla}@parisdescartes.fr

***Abstract*—The development of vehicular technologies and infrastructures leads to developments in mobility handling for wireless communications. Improving connectivity establishment and reliability became an issue, especially for vehicles that may move out of antenna coverage during connection establishment. This paper focuses on improving LoRaWAN connectivity for roaming devices by combining a machine learning predictor and exploiting DNS prefetching to gather information necessary for connection establishment before the device comes under coverage, thus reducing the overall latency for connection establishment. Extensive studies on various parameters such as antenna disposition or cache size.**

Keywords - IoT, V2I, V2X, DNS, Prefetching, Machine Learning Prediction, Traffic Prediction, LoRaWAN, Edge Networks

I. Introduction

The evolution of vehicular network generations led to new challenges in the different communication models such as V2V, V2I, V2P and V2X, where the connected vehicles can provide services like data caching or tasks offloading for other vehicles and also for users' devices. Besides, the high mobility of vehicles is an essential issue because it is related directly to the communication channel where the stability of the connection enhances the quality of service, particularly latency and delay.

From a user's point of view, access must be provided as smoothly as possible, without additional cost, to develop the technology's adoption. Furthermore, in roaming scenarios, serving all users as soon as possible would decrease the impact from other networks on their own gateways. From an operator's point of view, an increase in latency might incur congestion or gateway overload, which would decrease the Quality of Service for IoT solutions. Thus, reducing the impact of DNS requests when a device is joining becomes a key connectivity concern. It is particularly challenging in mobile environments.

Also, the issue behind storing and sharing data, or where to locate data caches, how long to keep DNS information cached and when to access it as an operator, is crucial to improve the network for backend mechanisms. Prefetching information is a common strategy to reduce network latency. Web browsers use such techniques to obtain IP addresses for domains within a web page, predicting that the user may click on a link, thus sparing the requests when a user clicks by performing the request beforehand.

Our proposal relies on a smart edge caching system that can pre-provision information based on device movement. A good example of underlying system to handle this data is DNS prefetching. The DNS is a crucial component on the Internet that fits the criteria for the data we wish to study. It might be used as a distributed database, and it could incur additional latency and hinder device connectivity when the system is too slow to answer.

This paper proposes and analyzes different data querying approach for mobile devices in an urban scenario through the example of DNS prefetching. We evaluate their impacts on both the user and operator's quality of service. We assume LoRaWAN connectivity. In a LoRaWAN scenario, DNS is involved in the roaming procedure. We assume that we can exploit DNS prefetching to query DNS servers based on device mobility to resolve device-specific information between the gateway and a DNS server. The prefetching can be as simple as requesting that nearby gateways prefetch the information (our algorithm **2** below) but could also rely on recent mobility models based on Machine Learning (ML) predictions (our algorithm **3**). This paper studies the consequences of prefetching information on antennas with regards to device mobility. In particular, we check if the information is prefetched adequately with respect to the actual vehicle location by observing the DNS cache hit rate and the number of DNS requests necessary to run the system (both prefetching DNS requests and on-the-fly DNS requests). We also study antenna cache occupation, cache hit-rate, DNS requests counts based on various parameters such as antenna deployment, vehicular density and cache size.

The presented use case focuses on provisioning DNS connectivity data necessary for the join exchange in a LoRaWAN connection establishment procedure, but this method applies to other data querying mechanisms. It is even more general since the strategies we propose to provision information in antennas for vehicular networks may be used for many other user data.

In the next part, we review the related works. Section III

presents our use case and expectations with this work. Section IV presents our algorithms (subsection IV-A) and our detailed scenarios (subsection IV-B). Then our results are detailed and discussed in Section V. Finally, our conclusions are summed up in Section VI.

II. Related Works

A. Improving communications using predictors in Vehicular Networks

In this part, we present existing works that resolve the communication failure problem caused by mobility in vehicular networks. Some of these solutions exploit machine learning techniques to predict device movement and improve communication efficiency.

Crisòstomo et al. [1] proposed a local route repair when the connected nodes in the network predict that the itinerary to the destination will break; they trigger the repair process. All nodes are equipped with an embedded GPS; besides, all packets contain node positions and other information. The limitations of the proposed approach are high resource utilization and the cost of using GPS.

Similarly, Goff [2] et al. proposed route maintenance that allows for creating a new alternative route before the break of the current communication link. The route is considered a broken route when the received signal power is close to the lowest detectable power; at this moment, a warning message is sent to the source node about the risk of a link break, Then the source node starts the process of finding a new route to reach the destination. The main limitation of this solution is when the source node does not receive the warning, which leads to a link failure during the communication.

In [3] authors proposed a protocol called PrAODV where a comparison is made between a threshold value and the signal power of the received packets; when the received signal is less than the threshold value, a ping message is sent from the current node to all its neighbors that must respond with a pong message, then the source node starts a route rediscovery. The main problem of this solution is the routing cost.

Similarly, in [4], authors proposed a route maintenance mechanism called PPAODV that enhances the QoS by predicting the risk of communication link failure. They used the Lagrange interpolation to predict if the current node will be in a dangerous zone in the next movement. Besides, when the next position is in a dangerous zone, a route repair process is triggered to find a new route to reach the destination, which consequently prevents the network from link break.

Vinod et al. [5] proposed a mechanism of link life prediction that creates an alternative link before it breaks. Vehicle mobility on a highway is used as a scenario to validate the results; besides, they used the microscopic or macroscopic (traffic flow, traffic density) approach to generate the vehicles' movements. The proposed algorithm uses vehicles' velocity and location to predict the route break.

Shelly et al. [6] proposed a statistic method for link lifetime in Vanet networks using an analytical model. They studied the impact of vehicle density, vehicle mobility, and the transmission range and analyzed the statistics of the communication link. Besides, they studied a case of two vehicles (A) and (B), where V_A, V_B and V_r are velocities of vehicle A, vehicle B and the relative velocities of pair of vehicles respectively.

Work in [7] proposed a link duration prediction via AdaBoost algorithm [8]. The proposed steps consist of aggregating the existing link metrics to generate many predictors; each predictor predicts if the link duration is under or over a threshold with high accuracy using the link metrics. In the next step, the algorithm determines the duration of the link using all the knowledge collected from these predictors.

Wang et al. [9] proposed a prediction model called extended link duration prediction (ELDP), which allows the vehicle to estimate the link duration with the other vehicles. Simulations in a city and highway show that the speed of vehicles has an impact on the link duration prediction in Vanet networks. In this work, a normal distribution needs to be used for vehicle speed.

Das et al. [10] proposed a network formation game called NGOMA algorithm for MAC-level re-transmission. It selects one node from the intermediate node, and in the case of a link failure, the formation game is used to select the relay node to re-transmit a packet from the source node to the destination node. The proposed algorithm reduces the delay and enhances the packet delivery ratio.

Similarly, Bhoi et al. [11] used a data forwarding technique to predict the link failure where a link existence diagram (LED) is generated to know the existing vehicles' links. The proposed techniques prove their efficiency in terms of end-to-end delay. Nevertheless, the GPS cannot detect obstacles and requires vast resources.

Authors in [12] proposed a route prediction in Vanet networks to resolve the problem of communication link failure; they proposed to use machine learning algorithms for prediction and then studied the efficiency of the proposed solution. Simulation results proved the efficiency of machine learning in route prediction compared to real vehicle mobility.

Each proposed solution improved the QoS in vehicular networks, especially solving the problem behind link failure during communications. Nevertheless, it is difficult to prove the efficiency of these solutions in dense networks with a vast number of vehicles. In addition, the impact of different obstacles is not studied in these works.

Some of these solutions exploit machine learning capabilities to predict device movements. Using artificial intelligence to support and predict device mobility can improve link quality and is more suitable for large-scale vehicular networks.

B. DNS performance, caching and prefetching

DNS prefetching relies on a prediction mechanism; the user could click on the link, so its browser performs the DNS query beforehand for all domains on a web page. This simple prediction mechanism can be transposed to any circumstances. [13] analyzed DNS traffic with the increase of IPv6 technologies in web hosting and put it in perspective with network traffic

increase in Japan and offered a prefetching-based solution to increase cache hit rate and reduce response times on web browsers. [14] proposes to study DNS queries in the context of web navigation (DNS over UDP requests) by studying when DNS queries are performed and when the information is needed. Their conclusion regarding prefetching is that no supplementary DNS cost applies thanks to prefetching. A good tutorial on prefetching and its consequences is provided by the Chromium project [15].

Fetching data using DNS comes with a short delay. [16] studied DNS responses with overall results outlining a 200ms response for 70% of their queries, and 90% of queries are realized within 1s. More recent analyses, such as [17] or [18] outline better results by combining anycast technologies and Content Delivery Networks for DNS. [17] studies responses from top resolvers which answer 90% of their requests within 100ms. Moreover, [18] provides additional information regarding DNS over TLS (DoT) resolution in which they outline failure rates with responses between 130ms and 230ms from top resolvers. Overall, the time inflation from additional security can be outlined around these values.

DNS over HTTPS (DoH) would add another supplementary cost up to 150ms as outlined by [17] measurements on public resolvers. Adding an integrity check with DNSSEC would increase the requests even further. Overall, sending two complete DNS requests completed with integrity check and secured with DoH would cumulate up to 1.1s of queries done within the first exchange between the ED and the RG. Our problem is as such: "Would it be possible to reduce that delay in a mobility context to reduce the impact from DNS querying on channel establishment?"

This is the reason why exploiting DNS to prefetch information is practical, as the information is queried either way; doing it beforehand, if possible, reduces the overall latency. Prediction algorithms help us determine where to provide the DNS information. This paper aims to analyze how we could reduce the overhead of DNS querying in mobility solutions for vehicular applications by studying various scenarios.

III. Motivation

This paper proposes DNS prefetching approaches to support LoRaWAN roaming connectivity through independent antennas. Having independent antennas requires a new join procedure each time a device changes its antenna, and the join uses DNS queries. In other words, to gather user data and provide network coverage in these scenarios, backend interconnection is necessary, and DNS serves as a facilitator in this handshake between servers, providing both discovery and security in the system.

Our previous paper [19] presented an approach to reduce the delay added by on-the-fly DNS queries necessary for device communication. We presented three approaches to handling DNS requests for vehicles. The first one is a classic on-the-fly DNS approach that serves as a reference: a device arriving under the coverage of a new antenna starts a join process, which requires DNS requests which are done at this time according to the traditional way. The second one uses proximity between antennas to decide which information to prefetch in advance (thus not on-the-fly): the DNS information the device will request is prefetched by the back end on the antennas neighboring the one covering the device before it arrives under their coverage. If a device is under the coverage of antenna A, the information is prefetched on all the antennas around antenna A so that when the device moves, it can find the information ready on its next antenna since its chance to arrive under the coverage of a neighboring antenna is high. This second approach is the state-of-the-art approach. The third approach is our proposal to exploit vehicular traffic prediction to forecast vehicle movement and prefetch the information accordingly onto the antennas: the information is prefetched where we expect the device to be in the near future, instead of duplicating it in every neighboring antenna.

We proved in [19] that DNS prefetching is an efficient tool to serve our proposal to reduce the delay added by on-the-fly DNS queries. Prefetching the information on nearby antennas, like in the second approach, can completely prevent DNS queries by performing them in advance around the closest antennas, but at a cost; more antennas realize the prefetching operation than the third approach, especially in a highly mobile environment. By exploiting recent ML capabilities for traffic prediction, we could provide a solution that heats the cache for 86% of requests, and that leads to a cache hit for 97% of them, on-the-fly DNS queries remaining necessary because of prediction failures for only 3% of queries. Overall, the ML system outperforms its nearby-activation counterpart in terms of antenna solicitation since only the most likely future gateways are provisioned: the second approach activates around 27% more gateways than the third.

Our previous study presented a preliminary performance study of a system with three possible DNS querying algorithms. The present paper aims to study the consequences of changing system parameters more extensively, such as antenna topology, vehicular density and adding a limit to the cache's size.

Adding these considerations to our study's scope would help us understand how the cache acts when extensively solicited. We propose another way to study antenna overload, sharper and with a better fit to the reality, while still considering a 5-minutes caching duration. We study cache congestion against the number of vehicles within the perimeter, considering a 5000 entry limit for the system while also increasing the number of vehicles within the vicinity of the antennas.

IV. Proposed approach and analysis methodology

A. Algorithm

LoRaWAN allows for two roaming techniques, passive roaming and handover roaming. Handover Roaming was introduced in LoRaWAN 1.1 specifications [20] and allows for roaming in similarly to cellular handover. Passive Romaing is older in LoRaWAN and has existed since LoraWAN 1.0.X. Passive roaming is the most developed roaming specification.

Passive roaming guidelines are detailed in [21], a document that will detail other roaming approaches in later versions.

We focus our study on the LoRaWAN passive roaming: it is the most developed, its specifications are detailed, and the solution is implemented in the state-of-the-art LoRaWAN implementation Chirpstack [22]. During passive roaming, a join procedure is triggered, introducing two DNS queries for channel establishment between the gateway and the backend infrastructure. The usual activation flow detailed on Figure 1 for roaming device consists in :

- an exchange between the End-Device and the serving Network Server associated with a nearby gateway
- a DNS query to identify the Join Server associated with the End-Device
- a DNS query to identify the home Network Server
- an exchange between Join Server and serving Network Server
- a response from serving Network Server to the device

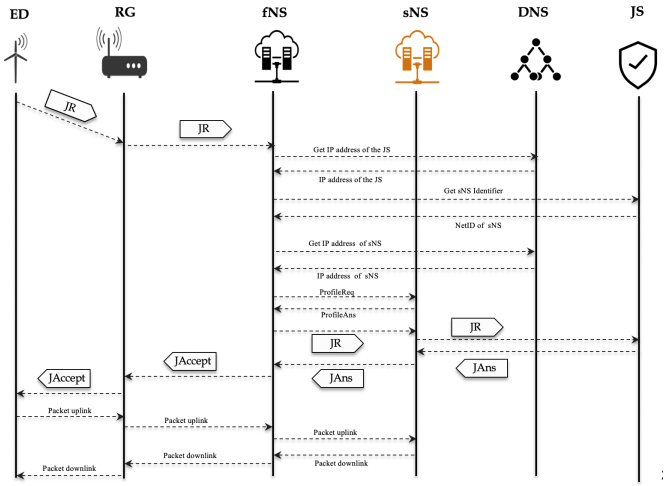

Figure 1: Usual LoRaWAN devices activation message flow

The idea of our method is to allow prefetching of the DNS information before needing it to save time during the activation procedure (cf. figure 2). The consequences of DNS prefetching on message flow are described in Figure 2; the information necessary to support the devices' connectivity is recovered before the device's Join Request; thus, the time corresponding to the various queries is saved from the first transmission and realized beforehand.

The activation flow (Figure 1) using prefetching becomes as described in Figure 2:

- a Prefetching DNS query to identify the Join Server associated with the End-Device
- a Prefetching DNS query to identify the home Network Server
- an exchange between the End-Device and the serving Network Server associated with a nearby gateway
- an exchange between Join Server and serving Network Server
- a response from serving Network Server to the device

The proposed algorithm, labeled as algorithm (**3**) from now, consists of periodically predicting a vehicle's future position to prefetch the DNS information necessary for the next in-roaming join if its covering antenna or network server (NS) changes. We assume each antenna change also corresponds to a NS change, which means that antennas provide independent network access, thus leading to a handover procedure. Formally, the proposed approach remains valid if NSs use several antennas, as long as the antennas of the city are shared between several NSs. Actually, the handover procedure is activated only when the device changes of NS.

We propose to use a Long Short-Term Memory (LSTM) algorithm to predict vehicle mobility inside the city [23]. Note that other prediction algorithms could also be tested. The mobility dataset consists of tuples: vehicle ID, date, time, and the position of vehicles (latitude, longitude). For each new predicted location, we survey the closest antenna and check if the device's information is available on the antenna's cache or should be queried. Actually, depending on the vehicle movements, DNS configuration (number of entries in cache, TTL, etc.) or antenna placement, it may come under an antenna's coverage where it has already been before.

Also, since the state-of-the-art solutions of prefetching are variants of what we call "nearby prefetching", we compare our approach to another reference algorithm: with the "nearby prefetching" algorithm, labeled (**2**) from now, the information to be prefetched is provisioned on all the neighbouring antennas around the current covering antenna. Of course, for each location, we check if the device's information is already available on the antenna's cache (if the device has already been there before) or should be queried before prefetching it. With this nearby prefetching algorithm, a moving vehicle is sure to find its information prefetched when it moves under the next antenna.

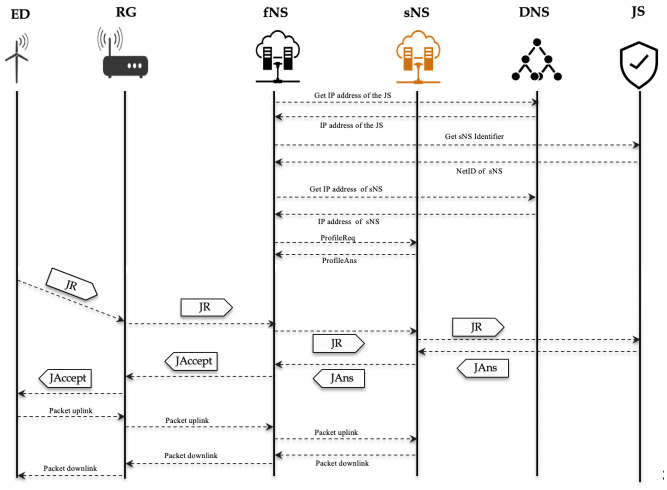

Figure 2: LoRaWAN devices activation message flow with our DNS prefetching mechanism

The main difference between the LSTM-based algorithm (hereafter called machine learning algorithm, or ML algorithm) and the nearby prefetching one is that the information is prefetched on all the neighbouring antennas in the case of the nearby one while it is prefetched only on the expected (predicted) next antenna in the ML one. At last, both algorithms are compared with the standard solution, labeled (**1**), where no prefetching is done at all: each time the device moves under the next antenna, it needs to query the DNS twice as usual. Of course, at the beginning of a vehicle trajectory, no prefetching can be done ; thus, the first location of the device is put on the side as "First DNS Query" and not taken into account in the comparisons of the algorithms.

B. Scenario, parameters and performance criteria

To validate our algorithm, we test it on an actual vehicle mobility dataset [24] in Rome city, Italy. The LSTM model is trained using the dataset. The data represents the real-time vehicle mobility for one month and traces the movements of 6992 devices within the Rome metropolis. Each vehicle is tied to 10 successive locations. Figure 3 shows part of the studied traces traced as a function of latitude and longitude.

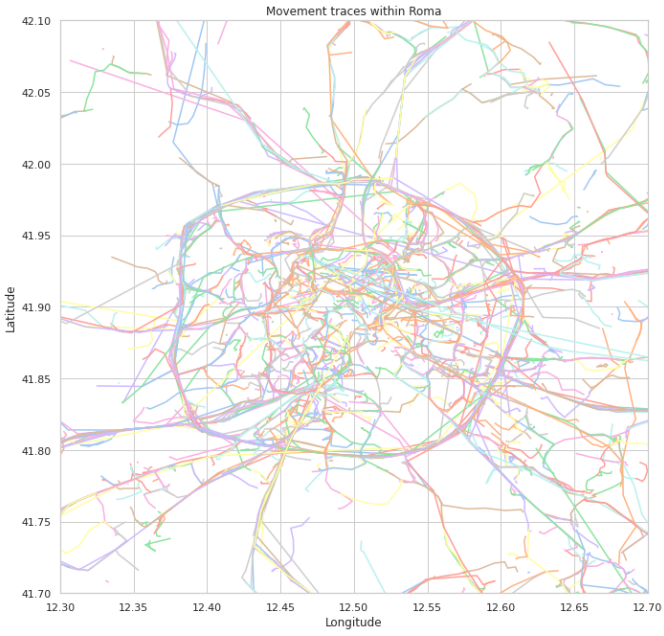

Figure 3: Vehicle mobility around Rome

We change several parameters to measure their impact on the algorithm performance: the antenna placement, the DNS cache size and the vehicle density.

There are different possible ways to vary antenna dispositions around the city. We consider regular and random antenna placements. For the regular antenna placement, we choose square (**a**) and hexagonal (**b**) topologies. For random antenna placement (**c**), we tried to mimic the real-life configuration where the radio resource is concentrated near the user. To this goal, the random antenna placement is realized by taking each trajectory independently, then placing an antenna with a 10% probability along this trajectory (thus obtaining 520 antennas, along the same values as regular placements). Once the decision to place an antenna is made by the random roll, the antenna is placed in a random position within a 3km disk around one of the vehicle's positions; thus obtaining a random antenna disposition with better coverage along major streets and highways within the city and its vicinity. We consider variable antenna coverage distance for random placement based on the number of neighbors it possesses. If the local antenna density is high, the LoRa cell size is reduced to take it into account, and the number of neighbors considered is thus reduced.

Considering proximity-based prefetching, we defined proximity for regular cases based on their neighbors in the placement pattern; that makes theoretically 6 neighbors in the hexagonal placement and 8 in the square placement. In practice, the border limit reduces this neighbor value to 5.66 for hexagonal placement and 7.17 for square placement. For the random placement, proximity is based on the distance between antennas. We consider various cell size based on antenna density :

- We decide on a 7-km wide cell size as a default value and count the number of neighbors in the random placement case.
- If the number of neighbors is more than the theoretical number for the square pattern (8 or more), coverage swaps to a mildly dense area and cell size is reduced to 4-km wide coverage. In this second case, we count the number of neighbors again.
- If this new number of neighbors is over the theoretical number for the square pattern again, coverage swaps to a highly dense area and cell size is reduced to a 1-km wide coverage. This last case is final, we count this final number of neighbors for antennas in highly dense areas.

Using this proximity determination algorithm, we end up with 5.17 neighbors per antenna average instead of the 55.86 neighbors per antenna average obtained considering only 7-km wide cells. This new 5.17 value observed is closer to the number for other scenarios than the 55.86 value obtained with fixed-size cells.

DNS cache congestion is not usually studied as it is out of the scope of classical DNS traffic issues, but our system caches a lot of DNS information, thus it is a subject of interest. We assume that each independent antenna will provide roaming access to devices within its reach. As described in figure 1, this means that the antenna will request the device's key from its home network (HN) and establish its connection to the ED thanks to them. We set the cache limit to 2500, which corresponds to a real cache size of 5000, considering that the activation flow from 1 processes two DNS requests. 5000 entries is close to the usual cache limit in DNS caches [25]. This allows us to study the consequences from adding a limit to the DNS cache on network traffic. Once the cache is full, the software would pop from the cache the oldest entry and put in a new one. We set a Time-To-Live (TTL) for our DNS

entries to 300 seconds; that means the DNS entries are kept in the DNS cache for a maximum time of 300 seconds. Two cases are studied : no-cache limit (**X**) and 2500-entries cache limit (**L**).

The vehicular dataset we use is traces from a sample set of vehicles in Rome, corresponding to 6992 vehicles. To simulate a more realistic traffic, we extrapolate it semi-artificially by duplicating 1000 times the trajectories at different times randomly chosen in the day. First, we duplicated 1000 times each sampled trajectory in the hour following it. This allows us to keep peak and off-peak hours. Also, to measure the sensitivity of the performance study to the traffic profile, we duplicated each sampled trajectory throughout the day. Each time a trajectory is duplicated, either for duplications in the same hour (**A**) or in the day (**B**), the corresponding duplications are uniformly placed either in the hour or in the day. Note that the most realistic traffic corresponds to the case where the duplications are done in the hour since the vehicular traffic is bursty with peak hours and off-peak hours, but the duplication in the day is only here for sensitivity analysis purposes. Thus, we moved to increase traffic density by duplicating the vehicle trajectories at different times along their paths. Using this method, we can consider multiple instances of vehicles along a given trajectory without generating paths from simulation software. Our given paths are known and can inform us about how the overall system operates when handling an increasing number of vehicles. Finally, this way, we simulate 6992000 vehicles.

Taking into account all the possible parameter combinations leads to 36 simulation combinations. We use these simulation results to try and define possible ways to improve the overall system for multi-tenant IoT deployments and try to point out essential backend considerations when working with these systems at scale. Finally, the simulation parameter combinations, summed up in Figure 4, are :

- Firstly, we consider the possible prefetching algorithms (No Prefetching, Proximity-based Prefetching, Machine-Learning-based Prefetching)
- Secondly, we consider possible ways to densify the traffic based on duplicating the trajectory within an hour or within the day.
- Thirdly, we take into account the possible antenna dispositions described above: Square, Hexagons or Random dispositions
- Finally, we add the cache limit and study the unlimited cache limit scenario compared to the usual size for DNS caches

Our criteria of interest are the number of uncached DNS requests performed by the whole system, with or without considering the prefetching requests, the TTL for the information in cache when the system is overloaded and the effective cache size measured for each antenna (can be studied as an absolute value or as a percentage of the defined maximal value of 2500).

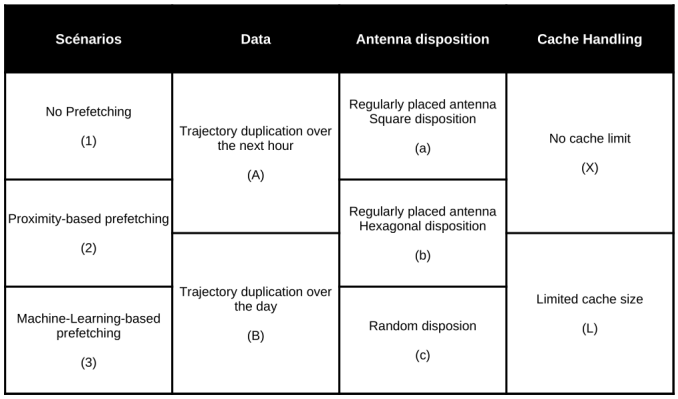

Figure 4: Scenarios distribution and labeling for all cases

V. Results & Discussion

In this section, to highlight the impact of one parameter against all the others, we aggregate all the values of the performance criterion of interest obtained by simulations except the parameter of interest. For instance, when estimating the cumulative distribution of the time averages of the cache sizes to look at the impact of the algorithm (cf. Figure 7), we mix in the same set the time averages of all the simulations with different antenna topologies and trajectory duplication modes, but an infinite cache capacity and a given algorithm (either (**1**), (**2**) or (**3**)). Another example is when comparing the number of DNS requests for each algorithm, we aggregate the numbers of on-the-fly DNS requests for all the scenarios (duplication ways, topologies, etc.) but we fix the data querying algorithms (**1**, **2**, **3**), one curve for each algorithm, without considering the other parameters' values. (Figure 5).

Since our mechanisms aim at reducing the number of on-the-fly DNS requests by a device, the primary performance criterion of interest is this number. Thus, it corresponds to DNS requests a device should query if the information is not found in the cache, for instance, due to a mobility prediction error. Figure 5 aggregates our results on the number of "classic" on-the-fly DNS requests processed by the system, but the algorithm ((**1**), (**2**) or (**3**)). This figure shows the cumulative distribution of the number of on-the-fly DNS requests. We observe a difference between the aggregated curves from our scenarios. The **2**nd and **3**rd algorithm are identical for the top 15% of requests and the ML algorithm (**2**) outperforms the standard solution (**3**) by around 3%. As expected, scenario **1** is the most efficient and outperforms the others in terms of number of DNS requests. This difference, once considered within the system, can be translated into time gains for device communication. Generally speaking, the smallest number of DNS requests is for the nearby algorithm, the highest one for the algorithm without prefetching. The ML-based algorithm is in between. It saves a lot of DNS requests but still needs some because of prediction failures.

Looking in more detail, it appears there are exceptions. Figure 6 provides the average number of usual DNS requests per device for each (non aggregated) scenario, following

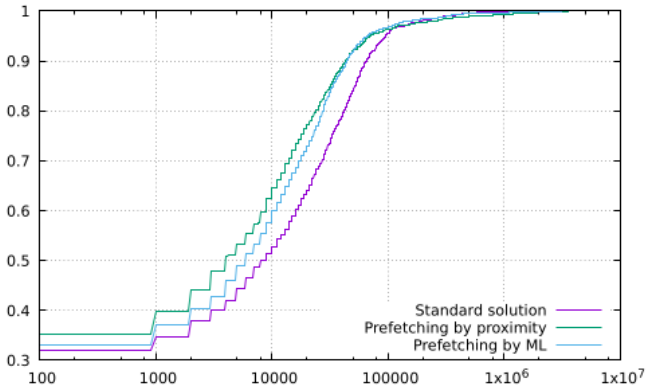

Figure 5: Number of DNS Requests for all cases grouped by prefetching scenario

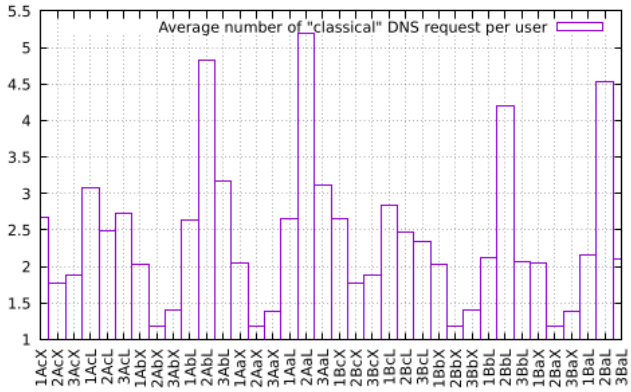

Figure 6: Number of DNS Requests per device for all cases

nomenclature summarized Figure 4. For all unlimited scenarios, we observe that prefetching scenarios outperform the no-prefetching ones. In contrast, considering a limit to the DNS cache, proximity-based prefetching creates an explosion in the number of DNS requests in the case of regular topology compared to the two other algorithms that appear close to each other.

Considering the usual limit to the cache size creates overloads within the system when prefetching comes within the scope, and, in that case, the information expires quickly (Figure 10). With regular antenna placement (cases **a** and **b**), this prefetching overloads the cache in a significant way and degrades the system's performance. This effect is less important when the antennas are located where the demand is actually (i.e. pseudo-random topology). For pseudo-random disposition, which is the most realistic of our cases:

- Without cache limitation, Machine Learning prefetching outperforms the no-prefetching scenario by 30%, and proximity-based prefetching outperforms the no-prefetching scenario by 33% independently of antenna disposition and traffic density.
- Considering cache limitation, prefetching gains introduced by Machine Learning over the standard solution is 11.7%, and prefetching gains from proximity are between 19.5% and 12.7%.

Overall, this comparison shows that for most cases, and especially the most realistic ones, our proposition to exploit machine learning capabilities outperforms the other scenarios and will lead to time gains for devices. Our machine-learning solution outperforms the no-prefetching scenario by adding this prefetching that provides the information beforehand and allows for time gains. Also, it outperforms the proximity prefetching case by mitigating the losses from traffic densification and preventing system overload. At last, it provides close results in scenarios where it loses compared to proximity.

Figure 7 describes the cumulative distribution of the average effective cache sizes for all scenarios, separated between our three querying algorithms: No Prefetching, Proximity-based Prefetching and ML-based Prefetching. The y-axis represents

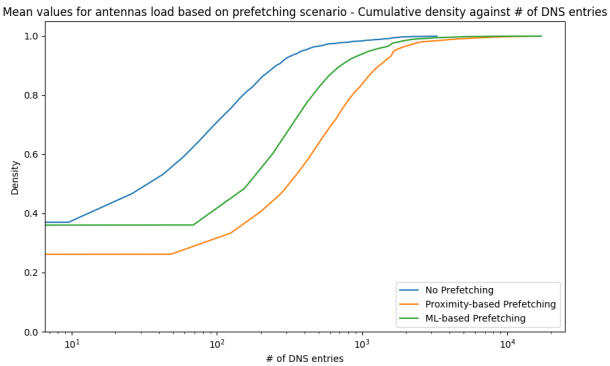

Figure 7: Mean values for antenna effective cache size - Aggregated scenarios

the cumulative distribution, and the x-axis is the cache size. We observe that the overall load required for all antennas without prefetching is the lowest, which means the cumulative distribution converges quickly to a lower value than in the other two cases. Without considering prefetching, which is the way the system in place works, the load seems manageable without significant issues with the DNS cache size. Proximity-based prefetching comes with a higher cost in terms of cache solicitation; the number of DNS entries in the cache is higher, with a slower progression and reaching a higher point up to more than five times the maximum value observed in the scenario without introducing prefetching. Machine Learning allowed us to mitigate this overcost from prefetching within some margin. Exploiting Machine Learning techniques leads to better results in terms of effective cache load than the Proximity-based scenario. These results lead us to think that Machine-Learning-based prefetching is efficient compared to Proximity-based prefetching. It would provide an interesting middle ground compared to a costly prefetching mechanism and classic DNS scenarios.

Figure 8 aggregates the results to illustrate the variation between regular antenna placements and pseudo-random antenna placement, and between Machine-Learning Based-Prefetching

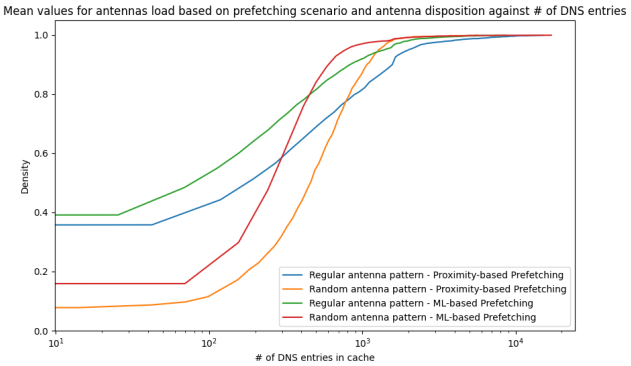

Figure 8: Mean values for antenna effective cache size - Scenarios aggregated on prefetching technique and antenna disposition

and Proximity-based Prefetching. For readability, we grouped our two regular patterns under a single curve, but both curves provide quasi-identical results. We observe with this scenario that antenna load is better distributed using machine learning than the usual proximity-based prefetching for both regular and pseudo-random placement.

With regular antenna disposition, we observe 38% of unsolicited antenna for proximity-based prefetching with progressive growth in load for the 62 remaining percent and overload in around 5% of cases. Machine Learning outperforms proximity-based prefetching with all antenna solicited between 2 and 10% less.

Pseudo-random disposition leads to a better load distribution with less antenna unsolicited (10 to 18%) and less antenna overloaded (2%). Once again, Machine Learning outperforms proximity-based prefetching with similar difference between both solutions.

Regardless of antenna disposition, we can say that ML-based prefetching reduces the system load compared to proximity-based prefetching by lessening the load on both low-charge antenna and the highly solicited ones.

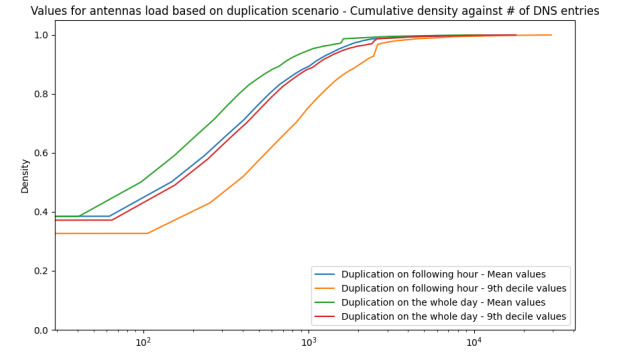

Figure 9: Mean and 9th decile values for antenna effective cache size - Scenarios aggregated on traffic densification

Figure 9 provides a comparison between cumulative distributions of antenna loads in terms of cache sizes for our duplication scenarios, aggregated for all results. It presents four curves: two compared mean values for antenna load (cache size) and two compared the 9th decile value, which gives an idea of the expected load considering a congested system. Regardless of the vehicular density, our machine learning proposal outperforms the proximity scenario. On the mean values curve, the daily distribution leads to less load than the hourly distribution. This result is more important on the 9th decile values as our dataset can be described as a heavy-tailed distribution.

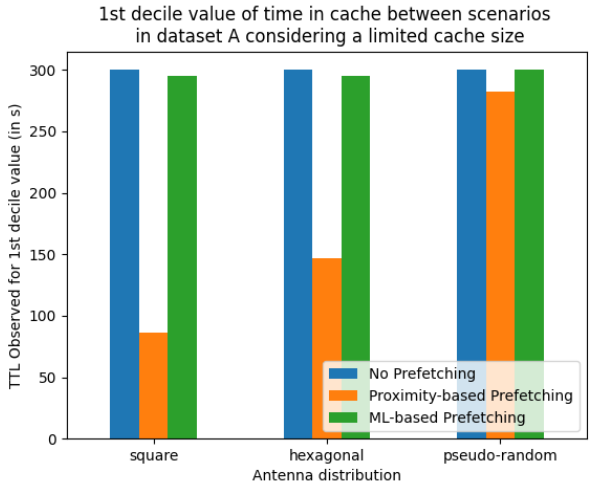

Figure 10: 1st decile values for DNS cache Time to Live in seconds for all scenarios - Dense traffic (A)

Figure 10 presents the 1st decile values for the TTL in our DNS caches for the higher density traffic (case A, duplicating trajectories over the hours). We observe that the overload for antennas, introduced by prefetching, leads to lowering the TTL for device identifiers within our DNS caches. This is expected since the DNS information in the cache is inserted in a pipe: the newest information replaces the oldest one when the cache is full. The first observation we can do is that for all antennas disposition, no-prefetching means no overload, and we observe a 300s TTL value. Then, introducing proximity-based prefetching degrades the system and can lower the TTL to the smallest values (86s in case a, 146s in case b and 282s in case c) in the DNS cache. Our ML proposal compensates for the loss from introducing additional information in the DNS cache by localizing the information efficiently. This efficient information localization leads to less overload on the antennas around the device by selecting the specific antenna that will need the information instead of provisioning the information on all the antennas around. The actual values observed for the 1st decile of data come from 86s to 295s in case a, from 146s to 295 in case b and from 282s to the maximal 300s value in case c.

As described at the beginning of the paper, reducing on-the-fly DNS queries leads to time savings. DNS query time is well

documented ([13] [14] [16] [17] [18]) and its usual cumulative introduced latency amounts between 60 ms and 250 ms; thus the queries can lead up to a full second of latency saving.

In terms of cache hit rate, the most efficient solution is the Machine Learning solution with 77% cache hit rate for all requests. Traditional DNS comes next with a 73% cache hit rate, and finally, proximity-based prefetching with 70%.

Proximity-based prefetching suffers a lot from cache limitation as the performances drop from 84% cache hit rate in unlimited cases to 56% with cache limit. As a comparison, traditional DNS drops from 75% to 71% and ML from 82% to 71%.

To summarize all our results, based on all these parameters, we can say that depending on the needs, the three techniques (traditional DNS **1**, proximity-based prefetching (**2**) and Machine Learning (**3**)) are an efficient tool to assist with device mobility.

If the issue is to reduce cache occupation, traditional DNS is the solution, it requires less storage on the antenna at the cost of more session establishment time[2222] during handshakes. The use of traditional DNS caching is already an efficient tool, especially with low-mobility devices.

If the issue is to reduce handover time in the more cases possible, proximity-based prefetching provides an efficient solution. By providing a systematic mechanism to make sure that the information will stick to device movement in all unlimited cache situations, proximity-based prefetching shuts down traditional DNS request times. This is nuanced when considering a limit to cache size, especially with regular cells, as shown in Figure 10, where proximity-based prefetching suffers from the limited cache size. Increasing the cache size from a 10-factor would solve almost all overload, but it depends on the number of devices in the vicinity. The solution works nonetheless in all less dense areas ; the issue with cache overload only concerns the top 20% antenna as the phenomenon completely disappears past this point.

Machine Learning provides a middle-ground between these two solutions, outperforming the proximity-based prefetching by around 10% in cache efficiency and almost suppressing the impact from limited cache size, at the cost of a 3.5 factor in overall DNS requests transmitted compared to the traditional DNS solution and leading to successful cache hit rate in around 77%.

VI. Conclusion

DNS prefetching is an efficient tool to reduce the delay added by on-the-fly DNS queries necessary for device communication. It is a way to prepare the information for the time when the vehicle will be under the umbrella of the right antenna. Prefetching the information on nearby antennas, like with algorithms **2** or **3**, can completely prevent DNS queries by performing them in advance. Our Machine Learning proposal with algorithm (**3**) is an improvement compared to the nearby-prefetching case **2**, allowing for similiar or better performances with a reduced cost.

Overall, the ML system would outperform its nearby-activation counterpart in antenna solicitation since only the most likely future gateways are provisioned: proximity-based prefetching leads to 2.25 times more requests overall from antennas than machine learning. The comparison keeps working when we develop on each sub-scenario, and machine learning outperforms prefetching for almost every criteria, including requests TTL (for which we observe up to 3 times less time in cache in algorithm **2** than **3**).

The only criteria in which proximity-based prefetching outperforms machine learning is the number of on-the-fly DNS requests when cache size is high (unlimited case). This leads us to discuss the pros and cons of overloading antennas and reducing device time loss.

Studying traffic densification, with our two different possibilities for traffic densification, leads us to think that the operation would work regardless of vehicular traffic and road disposition.

This paper focused on DNS prefetching, but the proposal can also be applied to any other environment for which anticipated data querying is useful such as video caching.

Acknowledging that exploiting machine learning predictions would be efficient and that pseudo-random distribution leads to better load distribution between antennas, it would be interesting to expand this work based on these two distributions.

Machine Learning algorithms can also be another parameter to study. We singled out a deep learning algorithm as part of this work, but various other traffic predictors exist and studying the impact of ML-predictors' performances on the overall system load would be interesting.

Finally, localized cache mutualization is also a possible point of interest for this work. We observe that most vehicles move in localized areas; thus, some information could be sent to exchange points mutualized for antennas in a given area to build a tradeoff between request processing and memory space.

VII. Acknowledgement

This work benefited from the support of the Energy4Climate Interdisciplinary Center (E4C) of IP Paris and Ecole des Ponts ParisTech. It was supported by 3rd Programme d'Investissements d'Avenir [ANR-18-EUR-0006-02]. This work was partly financed by the French National Research Agency through the CIFRE program [2018/0668].

References

[1] Sérgio Crisóstomo, Susana Sargento, Pedro Brandào, and Rui Prior. Improving AODV with Preemptive Local Route Repair. *Technical Report Series:DCC-2003-0, Universidade do Porto*, 2003.

[2] Tom. Goff, Nael Abu-Ghazaleh, Dhananjay Phatak, and Ridvan Kahvecioglu. Preemptive routing in adhoc networks. *Journal of Parallel and Distributed Computing*, page 123–140, 2003.

[3] A. Boukerche and L. Zhang. A performance evaluation of a preemptive on-demand distance vector routing protocol for mobile adhoc networks. *wireless communications and mobile computing*, 4:99–108, 2004.

[4] Sofiane Boukli Hacene, Ahmed Lehireche, and Ahmed Meddahi. PREDICTIVE PREEMPTIVE ADHOC ON DEMAND DISTANCE VECTOR ROUTING. *Malaysian Journal of Computer Science*, 19(2):189–195, 2006.

[5] Vinod Namboodiri and Lixin Gao. Prediction Based Routing for Vehicular Ad Hoc Networks. *IEEE Transactions on Vehicular Technology*, 56:2332–2345, July 2007.

[6] Siddharth Shelly and AV Babu. Probability distribution of link life time in vehicular ad hoc networks. *IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1418–1423, 2013.

[7] Jun Zhang, Meng Ying Ren, and Houda Labiod. Performance evaluation of link metrics in vehicle networks: A study from the cologne case. *ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications*, pages 123–130, 2016.

[8] Cao Ying, Miao Qi-Guang, Liu Jia-Chen, and Gao Lin. Advance and prospects of AdaBoost algorithm. *Acta Automatica Sinica*, 39(6):745–758, 2013.

[9] Xiufeng Wang, Chunmeng Wang, Gang Cui, Qing Yang, and Xuehai Zhang. ELDP: extended link duration prediction model for vehicular networks. *International Journal of Distributed Sensor Networks*, 12(4):5767569, 2016.

[10] Bhaskar Das and Jalal Almhana. A new cooperative communication algorithm for improving connectivity in the event of network failure in VANETs. *Computer Networks*, 128:51–62, 2017.

[11] Sourav Kumar Bhoi, Munesh Singh, and Pabitra Mohan Khilar. Predicting Link Failure in Vehicular Communication System Using Link Existence Diagram (LED). In *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*, pages 501–506. Springer, 2018.

[12] Mohammed Laroui, Akrem Sellami, Boubakr Nour, Hassine Moungla, Hossam Afifi, and Sofiane B Hacene. Driving path stability in VANETs. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2018.

[13] Kazunori Fujiwara, Akira Sato, and Kenichi Yoshida. DNS traffic analysis—CDN and the world IPV6 launch. *Journal of information processing*, 21(3):517–526, 2013.

[14] Mark Allman. Putting DNS in Context. In *Proceedings of the ACM Internet Measurement Conference*, pages 309–316, 2020.

[15] DNS Prefetching - The Chromium Projects. http://www.chromium.org/developers/design-documents/dns-prefetching.

[16] Jaeyeon Jung, Emil Sit, Hari Balakrishnan, and Robert Morris. DNS Performance and the Effectiveness of Caching. page 14, 11 2001.

[17] Austin Hounsel, Kevin Borgolte, Paul Schmitt, Jordan Holland, and Nick Feamster. Comparing the Effects of DNS, DoT, and DoH on Web Performance. In *Proceedings of The Web Conference 2020*, WWW '20, page 562–572. Association for Computing Machinery, 4 2020.

[18] Trinh Viet Doan, Irina Tsareva, and Vaibhav Bajpai. *Measuring DNS over TLS from the Edge: Adoption, Reliability, and Response Times*, volume 12671 of *Lecture Notes in Computer Science*, page 192–209. Springer International Publishing, 2021.

[19] Antoine Bernard, Mohammed Laroui, Michel Marot, Sandoche Balakrichenan, Hassine Moungla, Benoit Ampeau, Hossam Afifi, and Monique Becker. Prefetching of mobile devices information-a dns perspective. In *IEEE International Conference on Communications (ICC 2022)*, 2022.

[20] LoRa Alliance. LoRaWAN® Specification v1.1. https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/.

[21] LoRa Alliance. LoRaWAN TR009-1.0.0 Roaming Configuration Guidelines. https://lora-alliance.org/resource_hub/tr009-1-0-0-roaming-configuration-guidelines/.

[22] Orne Brocaar. Chirpstack, Open Source LoRaWAN solution. https://chirpstack.io.

[23] Mohammed Laroui, Aicha Dridi, Hossam Afifi, Hassine Moungla, Michel Marot, and Moussa Ali Cherif. Energy Management For Electric Vehicles in Smart Cities: A Deep Learning Approach. *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 2080–2085, 2019.

[24] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from https://crawdad.org/roma/taxi/20140717/taxicabs, July 2014. traceset: taxicabs.

[25] Daniel Aleksandersen. A survey of DNS caching and TTL in end-user client software. Blog post : https://www.ctrl.blog/entry/dns-client-ttl.html.