Mobility Prediction-Assisted Over-The-Top Edge Prefetching for Hierarchical VANETs

Zhongliang Zhao, Lucas Guardalben, Mostafa Karimzadeh, José Silva,

Torsten Braun, Susana Sargento

Abstract

Content prefetching brings contents close to end users before their explicit requests to reduce the content retrieval time, which is crucial for mobile scenarios such as vehicular ad-hoc networks (VANETs). In order to make intelligent prefetching decisions, three questions have to be answered: which content should be prefetched, when and where it should be prefetched. This paper answers these questions by proposing a vehicle mobility prediction-based Over-The-Top (OTT) content prefetching solution. We propose a vehicle mobility prediction module to estimate the future connected roadside units (RSUs) using data traces collected from a real-world VANET testbed deployed in the city of Porto, Portugal. We design a 3-tier caching mechanism and an OTT content popularity estimation scheme to forecast the content request distribution. We implement a learning-based algorithm to proactively prefetch the user content to VANET edge caching at RSUs. We implement a prototype using Raspberry Pi emulating RSU nodes to prove the system functionality. We also perform large-scale OpenStack experiments to validate the system scalability. Extensive experiment results prove that the system can bring benefits for both end-users and OTT service providers, which help them to optimize network resource utilization and reduce bandwidth consumption.

Index Terms

Content prefetching, vehicular ad-hoc networks (VANETs), mobility prediction, content popularity estimation, over-the-top services, road side units (RSUs), edge computing.

I. Introduction

In the past years, vehicular ad-hoc networks (VANETs) have evolved from a research topic to real-world deployments. VANETs provide car-to-car communications to address the increasing

Zhongliang Zhao, Mostafa Karimzadeh, and Torsten Braun are with the University of Bern, Switzerland. Lucas Guardalben, Jose Silva, and Susana Sargento are with the University of Aveiro and the Instituto de Telecomunicações - Aveiro, Portugal.

traffic demands for vehicle-correlated applications. Efforts have also been made to explore the potential of using vehicular network communications to improve transportation efficiency and to provide comfortable user experiences to passengers and convenience to drivers. The service characteristics of VANETs differ from conventional wireless communication services, which means that we have to analyze the specific requirements of various vehicles, their services, and applications. A recent study [1] highlights the importance of having a comprehensively coordinated system to meet the low-latency and high-mobility communication features for VANETs.

VANETs consist of hundreds of connected vehicles, which use DSRC (Dedicated Short-Range Communications), Wi-Fi, or cellular to connect to each other and to the infrastructure. A real-world VANET testbed includes the installation of communication systems, which assure vehicle-to-vehicle (V2V), vehicle-to-roadside (V2R) and vehicle-to-infrastructure (V2I) communications. A VANET is deployed by installing on-board units (OBUs) on vehicles, road side units (RSUs) at particular city spots, and by connecting the RSUs to the access infrastructure (copper or fiber network).

Over-the-top (OTT) services have grown in recent years. The number of OTT services, characterized by being transmitted over any network without the operators' control in the distribution process, has been increasing. Supported by an existing and usually free Internet backbone, service providers have exploited the characteristics of unmanaged networks to deliver services to their consumers without having to invest heavily in the infrastructure. Due to their nature, OTT services have an inherent widespread reachability and they are able to accommodate virtually any Internet-connected device, without requiring network-specific equipment or management capabilities [2].

The rising popularity of multimedia streaming protocols has led to its widespread adoption in modern multimedia services, such as YouTube, Netflix, etc. As shown in Section 2.4, this set of streaming protocols exhibits very specific traits, which depend on the streaming protocol in use. For example, delivering content using adaptive streaming protocols over HTTP, although conceptually similar to delivering any other web content, presents a set of challenges regarding scalability and maintenance of adequate QoE levels, especially in highly dynamic environments, such as it is the case for VANETs [3]. Since Internet Service Providers (ISPs) allow third-party services to use their network for free, this type of service is called Over-the-Top, and content is delivered without direct operator management. One emerging OTT application scenario is vehicle on-board video streaming, and many works on OTT service and content prefetching for VANETs have been proposed [4].

A typical VANET use case is the one where mobile users are downloading multimedia streaming services from the connected RSUs while travelling in a car. To ensure seamless content delivery, an efficient OTT content prefetching mechanism is required to improve the quality of experience of mobile users in moving vehicles. To address the problems of where to prefetch the OTT content and which content should be prefetched in VANETs, the system should be able to know the content popularity and the future locations of vehicles. This leads to the problems of vehicle OTT content popularity estimation and vehicle location prediction. Considering the specific characteristics of VANETs, both problems should be resolved while being subject to the VANETs features, such as short V2V/V2R contact time, fast node movements, and unreliable radio channel characteristics in vehicular communications environments [5].

Given the challenges of VANETs, such as short V2R contact times, we propose a vehicle mobility prediction-assisted OTT prefetching mechanism. The system estimates the OTT content popularity and performs content prefetching operations based on the predicted future connections of RSUs and vehicles. The contributions of the paper can be summarized as follows:

- We propose a hierarchical VANET architecture, which supports content caching at different layers. The 3-tier architecture enables vertical cache aggregation, which optimizes the caching operations and improves the system performance.
- We provide a deep exploration of a rich vehicle dataset collected from a real-world VANET testbed deployed in the city of Porto, Portugal. The testbed consists of 600+ bus/garbage trucks as OBUs, and 70+ RSUs. This dataset empowers us to understand the real OBU-RSU connectivity, which enables us to develop a practical VANET prefetching mechanism in real implementation using IEEE 802.11p compatible hardware. Plus, it makes our work different from existing works that are simulation-based.
- We design a content prefetching mechanism assisted by vehicle mobility prediction. The proposed mobility prediction solution is a dynamic Markov chain-based model, which adaptively selects the first- or the second-order Markov chain model based on the available trace quality. We implement a context-aware content popularity estimation mechanism for VANET video streaming services.
- We implement a system prototype using the Raspberry Pi platform emulating the RSU to perform real prefetching operations in a small-scale network. We use the OpenStack infrastructure to emulate a large-scale scenario with 40 RSUs to validate the system scalability. Results from extensive experiments prove the capability of the system.

The rest of the paper is organized as follows. Section II describes the related work. Section III elaborates the design details of system architecture and the subcomponents. Section IV includes implementation details of both a small-scale prototype and a large scale OpenStack experiment. Section V discusses the results of the prototype and OpenStack experiments. Section VI concludes the paper and foresees future works.

II. Related work

VANET-correlated experiments should be done using IEEE 802.11P/WAVE compatible hardware mounted on vehicles running on real roads. However, due to the prohibitive costs of real-world VANET deployments, most of the research activities on VANETs focus on simulation-based study or very small-scale testbeds. The Veins framework [6] was proposed as a hybrid simulation framework to develop efficient Inter-Vehicle Communication (IVC) protocols for on-demand route planning and safety applications. Authors of [7] proposed another simulation framework, which includes driving, traffic, and network simulators, for testing and evaluation of cooperative intelligent transport systems applications. C-VeT [8], an open VANETs testbed within the UCLA campus, integrates a wireless mesh network solution. HarborNet [9] is a real-world testbed for research and development in vehicular networking that has been deployed successfully in the seaport of Leixões in Portugal. It allows cloud-based code deployment, and distributed data collection from OBUs and RSUs.

In a VANET deployment, vehicles could download content from RSUs when the OBU-RSU association is established. However, due to the high speeds, the actual contact time between the vehicle and RSU is very short. For instance, a moving car with an average speed of 50 km/h and equipped with the IEEE 802.11p Wireless Access in Vehicular Environments (WAVE) device with a transmission range of 200 meters has a contact time with a connected RSU of less than 30 seconds. Moreover, [5] has evaluated the performance of the WAVE protocol, and it claimed that in dense and high load scenarios, the network throughput is decreased while the delay is increasing significantly. Therefore, it is important to have the content to be downloaded available when the car is approaching the RSU, such that the short vehicle-RSU contact period can be fully dedicated to content transmission. This leads to the design of efficient content prefetching mechanisms at the VANET edge of RSUs. Several challenges arise when trying to bring content as close as possible to the vehicle consumers, such as the lack of memory and the limited processing power of equipments to store them (e.g., embedded cache on OBUs and

RSUs), etc. Therefore, in order to optimize multimedia content delivery in vehicular networks, prefetching strategies must be able to reduce the user-perceived latency by predicting next OBU-RSU connection to store OTT contents in advance [10].

Therefore, a prefetching mechanism needs to be used in conjunction with smart caching strategy. To be effective, the prefetching operation must be performed timely, to be useful and with low overhead. The work of [11] proposes a model to take into account the evolution of content popularity, which is used by a caching algorithm to keep track of the user requests. However, the impact of the dynamic model building overhead is not considered in the experiments. In [12], a predictive approach is taken towards content popularity, where data traces are used to build synthetic Gaussian, exponential, and power law models, which are then used in a modified least frequently used (LFU) caching policy. The work in [13] focuses on prefetching content to the clients' devices. Caching and prefetching are highly intertwined as both require a deep understanding of the content characteristics, and work towards the goal of reducing peak bandwidth consumption. Therefore, a content distribution network in vehicles should consider the particularities of every content object that will be cached [14].

Very recently, researchers started to analyze the possibilities of applying vehicle mobility prediction information to improve the proactive caching performance for VANETs. Many works are based on the simulation using the San Francisco taxi cabs mobility dataset [15], which contains GPS coordinates of approximately 500 taxis collected over 30 days in the San Francisco Bay Area. [16] presented a proactive caching strategy that leverages Information-Centric Networking (ICN)'s flexibility of caching data anywhere in the network, rather than only at the edge. They evaluated their protocol by performing simulation using the San Francisco taxi cabs mobility dataset. To estimate OBU-RSU connections, they assume that multiple RSUs (with known GPS coordinates) are distributed in San Francisco such that they can calculate the OBU-RSU connections based on distances. [17] proposed a second-order Markov chain based vehicle mobility prediction solution for content prefetching in VANETs. They also conducted simulation studies using the San Francisco dataset, and used a wireless AP coverage database in San Francisco to estimate the AP-vehicle connections. However, this AP coverage database is not designed for VANET applications intrinsically, which also limits their contributions. [18] builds a model of vehicle path assignment to estimate the vehicle's moving path and its sequences of visited APs along the path. However, as authors indicated in the paper, they ignored the time for vehicles to discover and associate with APs.

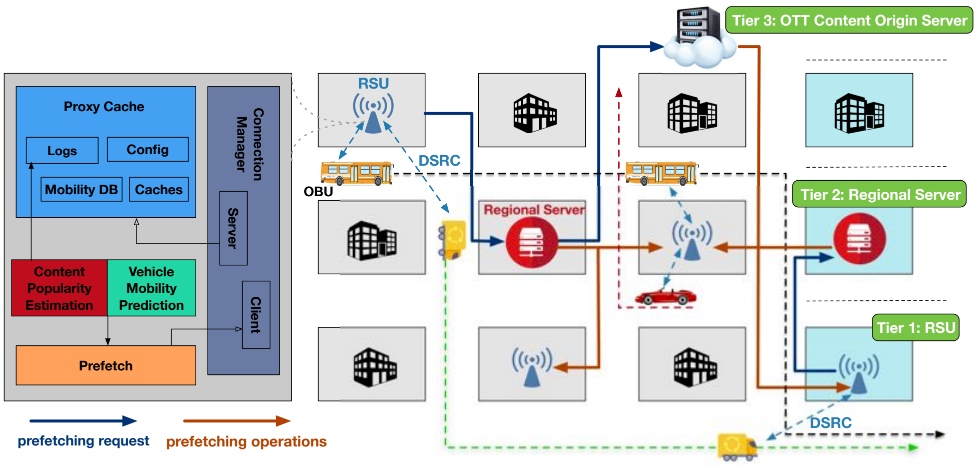


Figure 1: System Architecture.

Therefore, this work focuses on the practical design of a VANET OTT content prefetching solution by using the real OBU-RSU connectivity data collected from a real-world VANET testbed. We propose a hierarchical VANET architecture, which includes smart caching at different tiers. The system includes a vehicle mobility prediction engine, which estimates the future connected RSUs of an OBU, such that the content can be prefetched to the correct place. A content popularity model is also proposed to estimate the future content requests. The design has been validated in both a real prototype and OpenStack experiments.

III. System Models

In this section, we describe the architecture of the proposed mobility prediction-assisted OTT content prefetching mechanism in a hierarchical VANET.

A. System Architecture

The proposed system architecture is shown in Figure 1, which includes a 3-tier hierarchical caching design. The architecture includes three main entities: RSUs, regional server, and origin server. Their functions are explained as follows:

- *RSUs* are tier 1 edge cache units deployed on road sides that provide requested content to mobile vehicles when they are connected.
- *Regional Servers* work as tier 2 aggregation caches. They answer the prefetching requests from the RSUs that are deployed within their regions, and prefetch the requested content if available in their local cache to the indicated places. Otherwise, they forward the request to the origin server. They also include some logics to optimize the prefetching decisions.

- *Origin Server* works as tier 3 cache, and it stores all the content and chunks. It responds to prefetching requests from regional servers.

As illustrated in Figure 1, two regions (marked as light grey blocks and light blue blocks) are served by multiple RSUs, which are covered by two regional servers. *OBU devices* are installed on vehicles, such as buses or garbage trucks. The 3-tier caching architecture has many benefits. When a vehicle changes the associations from one RSU to another one, its content objects will be cached on the previous edge cache and also on the tier 2 aggregation caches. When asking for the same content once it is connected to the new RSU, there is no need to request the same content from the origin server, and the request can be served directly from the tier 2 aggregation cache. In the event of an edge cache server failure, the aggregation cache is also helpful to rebuild the cache of the backup edge caches, and there is no need to bother the origin server. The 3-tier aggregation cache is also useful when a Content Delivery Network (CDN) has a high geographical diversity and potentially large access delays to the origin server.

Each RSU node consists of five internal components. The *Connection Manager* manages the DSRC connection between RSU and vehicles. It also interconnects RSUs with the regional server to send the prefetching requests and receive content. The *Proxy Cache* is responsible for managing the context information about the connections of the RSU with the vehicles. It has an internal cache to store the content, and a database of historical movement traces of vehicles, which are needed in the location prediction. The *Content Popularity Estimation* predicts the content that will be requested by the mobile consumers. This solves the problem of ”what to prefetch”. The *Vehicle Mobility Prediction* module is responsible to predict the future connected RSUs and the corresponding connection probabilities of the vehicle. It answers the question of ”where to prefetch”. The *Prefetch* module takes the results of the *Content Popularity Estimation* and *Vehicle Mobility Prediction* modules, and sends the prefetching request to the regional server. The origin server stores all the content chunks. Moreover, it has two more data records:

- *Content Metadata* contains information that is associated with each media element (the core elements representing basic media types). In Video-On-Demand (VOD) services, content metadata includes content title, Electronic Program Guide (EPG) information, such as the original broadcasting program date, broadcast station, program title, episode number, series identifier, duration, etc. In case of live content, the origin server may act as conventional HTTP server, delivering the live content over many sources (e.g., HTTP-based live stream-

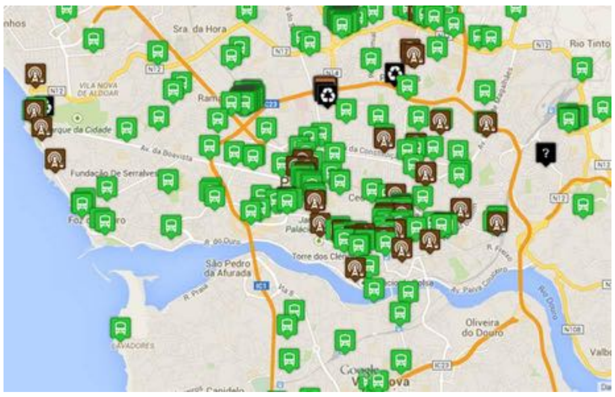


Figure 2: Real-time locations of OBUs (green nodes) and RSUs (brown nodes) provided by GPS of the VANET testbed deployed in the city of Porto, Portugal.

ing, Mpeg-dash, etc.). An example of live streaming metadata information includes a timestamp of the last video recorded, the width/height of the video in pixels, bit rate, frames per second, duration, etc.

- *Demand Logs* provide traceability by recording who requested what content and when. Therefore, it provides timestamped records that map user requests to VoD programs and the type of the streaming for live content.

B. Real-World VANET Dataset

This work considers real bus traces collected from the VANET testbed deployed in the City of Porto from October 2016 until August 2017 [9], [19] (as shown in Figure 2). This urban-scale testbed consists of 600+ networked vehicles using IEEE 802.11p, WiFi or 4G to connect to each other and to the infrastructure through 4G as well as 70+ RSUs scattered along the city. GPS positions and accelerometer data are gathered from vehicles' OBUs. The data trace collection process basically is done by a deployed DTN (Delay Tolerant Network) system, which makes use of the existing vehicular network infrastructure, communicating through V2V DSRC WAVE technology to carry and forward data towards its final destination. All information is stored in a MySQL database, which has available logs from the OBUs' positions and speeds. The positions log database stores all the collected raw data, which includes three tables:

- *RSUs Traffic Information* stores information about the amount of data traffic transmitted via different technologies (4G, WiFi, DSRC, etc).
- *OBUs Information* stores information about the OBU location, speed, connected RSU_ID, number of hops to the data server collector, 4G/DSRC traffic, etc.
- *RSUs Information* stores information about the RSUs in terms of the download/upload traffic volumn;

This work focuses on the OBUs_Information table, which includes valuable information for vehicle location prediction. The GPS time identifies the exact time when the spatial attributes (latitude, longitude, altitude, heading) were captured, which are used to calculate the average speed of OBU nodes. With the stable data collection, the VANET testbed collects 1.2 up to 3 millions of SQL records per day. However, if the VANET link conditions are bad or with temporary equipment failures, the amount of data collected will be greatly reduced. Therefore, based on the amount of collected data during the data collection period, we classify user trace data into two categories: vehicles with an average number of 500k entries of data in the database are considered as with good quality, and vehicles with an average number of 250k entries of data are considered as with bad quality.

C. Content Popularity and Prefetching

The management and control of edge caches (RSUs) are performed mostly using proxy-based cache solutions. In our model, the proxy cache plays an important role, and it helps to orchestrate where, which, and how the vehicles (OBUs acting as video consumers) are watching adaptive segmented HTTP-based delivery videos. When deployed in a RSU, the proxy cache deals with a small cache storage and it can be overloaded quickly, leading the caching strategies to replace more content objects. Therefore, the prefetching model should have high precision on choosing the correct content chunk to be cached on the correct RSUs, such that excessive back-end traffic to the origin server can be avoided, and the content delivery time can be reduced.

Given the characteristics of multimedia transmissions, one important thing to consider is which chunk will be more popular for future client consumption such that the specific chunk should be populated on the closest RSU cache before the vehicle's request. For adaptive segmented HTTP-based content delivery, such as Dynamic Adaptive Streaming over HTTP (DASH), the prefetching technique should involve the prediction of the popularity of future chunks of a given video in a certain quality, and populate the RSU cache to reduce the hit misses. Some additional features should also be considered when designing a prefetching model in VANETs, such as the intermittent wireless connections between OBU-RSU, network congestions, low bandwidth available, and the video popularity.

The proposed prefetching strategy analyzes the behavior of each OBU consumer and is based on the current quality and the last $n = 4$ requested chunks. A simple linear regression model is used to predict which quality will be requested next, as explained in Equation 1.

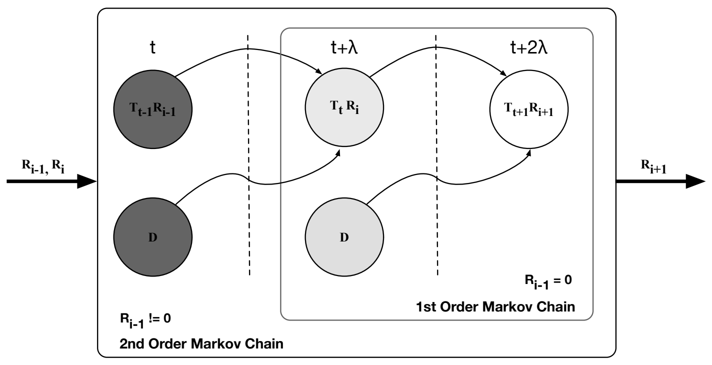


Figure 3: Hybrid Markov Chain.

$$\begin{aligned} X_{(t+1)} &= w1 \times X_{(t)} + w2 \times X_{(t-1)} +, \\ & w3 \times X_{(t-2)} + w4 \times X_{(t-3)} + w5 \times X_{(t-4)} \end{aligned} \tag{1}$$

$X_{(t+1)}$ is the expected quality, $X_{(t)}$ is the current one and $X_{(t-1)}, ..., X_{(t-4)}$ are the last 4 qualities. From our previous experience [20], a value of 4 past records is good enough to infer the next quality to be consumed as well. Basically, the vehicle content consumption pipeline is as follows: i) the vehicle asks for a chunk; ii) the RSU acts as proxy-based cache answering the request back; iii) the prefetching model predicts which quality and where the next chunk will be placed, and iv) the RSU as proxy-cache then caches the predicted content.

D. Vehicle Mobility Prediction

This section presents the proposed vehicle mobility prediction algorithm. The approach is based on a hybrid Markov chain model, which adaptively selects from the first-order or the second-order Markov chain, depending on the availability and quality of vehicle' traces.

The proposed hybrid Markov model benefits from both the first-order Markov chain presented in [21], and the second-order Markov chain. The rationale behind using a hybrid model is that the standard first-order Markov chain models are memoryless predictors, which means that the next location (vehicle's connected RSU) depends on (*i*) it's current location, (*ii*) the current time, and (*iii*) the day of the week of the movement. The second order Markov chain model is slightly different, as the prediction considers not only the current state information, but also the one from the previous state. The utilization of two state information could increase prediction accuracy. However, when trace data includes discrete gaps, the 2-order state transition conditions will not be met, which will lead to poor performance for the second-order predictor.

The proposed hybrid model is illustrated in Figure 3, in which a Markov chain state consists of a time step and a RSU ID. Equation 2 defines the calculation of future location probability,

Table I: Vehicle mobility prediction algorithm parameters

Parameter Name	**Parameter Definition**
R_i, OBU_i	RSU ID i, OBU ID i
D, T_i	The day of the week, and time of the day
λ	Future time interval
$C = \{R_1, .., i\}, \|C\| = I$	Set of RSUs
$U = \{OBU_1, .., j\}, \|U\| = J$	Set of OBUs
$N_{R_i}(t)$	Number of OBUs connected with RSU R_i at time *t* (*e.g.,* m)
$N_{R_i}(t+\delta)$	Number of OBUs in RSU R_i at time *t*+δ (*e.g.,* M)
n_1	Number of OBUs that may be connected with R_i at time *t*+δ
n_2	Number of OBUs that may be disconnected from R_i at time *t*+δ
$\triangle m = M - m = n_1 - n_2$	The difference between the number of OBUs in RSU R_i at time *t* and *t*+δ
$F_{R_{i'}(t)}$	Subset of all OBUs out of R_i at time *t*
$F_{R_i(t)}$	Subset of all OBUs in R_i at time *t*
P_{j_1} , OBU_{j_1} ∈J	$P\{OBU_{j_1}\ is\ in\ R_{i'}\ at\ time\ t\} \times P\{OBU_{j_1}\ moves\ to\ R_i\ at\ time\ t+\delta\}$
P_{j_2} , OBU_{j_2} ∈m	$P\{OBU_{j_2}\ is\ in\ R_i\ at\ time\ t\} \times P\{OBU_{j_2}\ moves\ from\ R_i\ at\ time\ t+\delta\}$

in which R_i represents a RSU with ID i, D indicates the day of the week (e.g., Saturday), T_i defines the time of the day D, and λ determines the future time interval. Table I shows the parameters used in the prediction algorithms. As we can see from Equation 2, the conditional distribution of the first-order Markov chain only depends on the current location (*RSU*) of the vehicle, time and the week day. But the second-order Markov chain comprises both current and previous visited states, time and weekday.

The first-order and second-order Markov chain models of Equation 2 can be further decomposed into a location-dependent component and a time-dependent component, which represents a location-dependent distribution, and a time-dependent distribution (as shown in Equations 3 and 6). For the second-order Markov chain, as we can see from Equation 6, both the location dependency and the time dependency distributions are based on the current and previous states. The location distribution can be modelled as a Mobility Markov Chain (MMC) that encodes the frequency (*probability*) of transitions between the RSUs. In order to derive the transition probability matrix of the Markov chain, we calculate the probability of moving from one RSU to other(s) for each individual vehicle, by counting the number of transitions from one RSU to another on the given weekdays (e.g., Mondays). For the case of the second-order Markov chain, the counting of transition frequency happens only when the vehicle's movement is continuously following the sequence of two states.

$$Pr(R_{i+1}(t+2\lambda)) = \begin{cases} Pr(R_{i+1}|R(t+\lambda)=R_i, D, \\ \qquad\qquad T(t+\lambda)=T_i) & \text{if } R_{i-1}=0 \\ \\ Pr(R_{i+1}|R(t)=R_{i-1}, R(t+\lambda)=R_i, \\ T(t)=T_{i-1}, T(t+\lambda)=T_i, D) & \text{if } R_{i-1}\neq 0 \end{cases} \tag{2}$$

$$Pr(R_{i+1}|R(t+\lambda)=R_i, T(t+\lambda)=T_i, D) \tag{3}$$

$$=Pr(R_{i+1}|R(t+\lambda)=R_i) \tag{4}$$

$$+Pr(T(t+\lambda)=T_i, D) \tag{5}$$

$$Pr(R_{i+1}|R(t)=R_{i-1}, R(t+\lambda)=R_i, T(t)=T_{i-1}, T(t+\lambda)=T_i, D) \tag{6}$$

$$=Pr(R_{i+1}|R(t)=R_{i-1}, R(t+\lambda)=R_i) \tag{7}$$

$$+Pr(T(t)=T_{i-1}, T(t+\lambda)=T_i, D) \tag{8}$$

$$\begin{aligned} P\left\{N_{R_i}(t+\delta)=M\right\} = \sum_m P\left\{N_{R_i}(t+\delta)=M \mid N_{R_i}(t)=m\right\} \times \\ P\left\{N_{R_i}(t)=m\right\} = \sum_m \langle \sum_{n_1,n_2,n_1-n_2=\triangle m} P\left\{N_{in,R_i}(t+\delta)=n_1\right\} \times \\ P\left\{N_{out,R_i}(t+\delta)=n_2\right\}\rangle \times P\left\{N_{R_i}(t)=m\right\} \end{aligned} \tag{9}$$

$$\begin{aligned} P\left\{N_{in,R_i}(t+\delta)=n_1\right\} = \sum_{A_1\in F_{R_{i'}(t)}} \prod_{j_1\in A_1} P_{j_1} \prod_{j_1'\in A_1^c} \left(1-P_{j_1'}\right) P\left\{N_{out,R_i}(t+\delta)=n_2\right\} \\ = \sum_{A_2\in F_{R_i(t)}} \prod_{j_2\in A_2} P_{j_2} \prod_{j_2'\in A_2^c} \left(1-P_{j_2'}\right) \end{aligned} \tag{10}$$

From the probability distributions of the OBU's future connected RSUs, we can further estimate the number of connected OBUs to a given RSU at a future moment. Therefore, next, we target at predicting the probability distribution of the number of connected OBUs with a specific RSU within a given time. This information will help us to optimize the caching strategy for content objects that might be requested by multiple vehicles that will be connected to the same RSU simultaneously or following a continuous time series. For instance, if we know that two vehicles will be connected to the same RSU at time t and $t + \lambda$ and they will request the same chunk of content, the targeted RSU should keep that chunk for both vehicles at t and $t + \lambda$. Parameters used in the probability distribution estimation of the number of OBUs connected to a RSU is also shown in Table I. The probability distribution of the number of OBUs that will be connected to a specific RSU is calculated at the regional server side. This is because this calculation requires the knowledge of an OBU's future connected RSUs, which is generated once the relevant RSU receives a request from a connected OBU. Therefore, whenever a RSU generates a transition probability matrix for an OBU, it should notify this to the regional

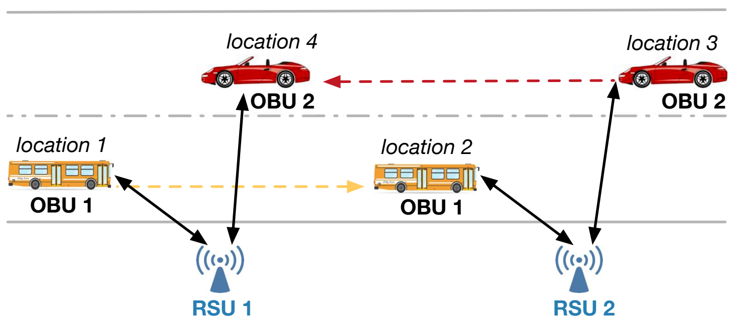
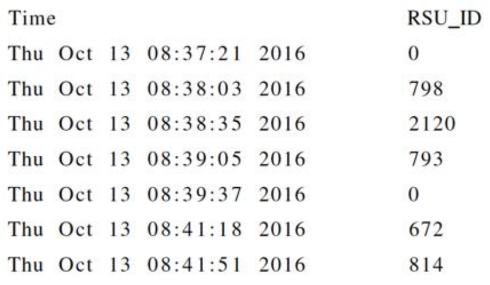


Figure 4: OBU's connections with multiple RSUs

server such that the regional server is updated about the OBU's future connected RSU. When the regional server is aware of the information about the probability distribution of the number of OBUs that will be connected to a specific RSU, it will optimize the caching at the relevant RSUs. In Equation 9, $P\left\{N_{in,R_i}(t+\delta)\right\}$ and $P\left\{N_{out,R_i}(t+\delta)\right\}$, describe the probability of n_1 OBUs that may be connected with RSU R_i and n_2 OBUs that may be disconnected with R_i at time $(t+\delta)$, respectively. These probabilities can be calculated using Equation 10.

Apart from the theoretical reasons in favour of a higher order Markov chain, practical experiences gained from empirical studies also lead us to favour a higher order model [22]. As shown in Figure 4, when both *OBU 1* and *OBU 2* are connected with *RSU 1* simultaneously at *location 1* and *location 4*, it is not possible for *RSU 1* to predict OBUs' next locations with only their current location information. However, if we also know that in the next time slot, *OBU 1* is connected with *RSU 2* at *location 2* or in the previous time slot, *OBU 2* was connected with *RSU 2* at *location 3*, then we can infer their movement directions, which will enable us to distinguish them and predict their future locations with higher accuracy. Therefore, for the vehicle mobility prediction problem, a higher order Markov model is definitely better than the memoryless first order Markov model. However, this statement is based on the assumption that there will be enough information about the vehicle movement data available. As we show later in the evaluation part, we can see that if the dataset is of poor quality, then the second-order Markov chain model will perform much worse than the first-order Markov chain. This is the reason why we design a hybrid Markov model, which adaptively selects from the first-order or the second-order Markov model based on the data quality.

As learned from our previous experience [23], trace quality has crucial impact on the prediction performance. Therefore, we first classify the dataset into two groups (good or bad quality) based on the number of recorded instances during the whole data collection periods. We choose data from four vehicles with good quality of trace data (e.g., with 460000-500000 instances of SQL



Time	RSU_ID
Thu Oct 13 08:37:21 2016	0
Thu Oct 13 08:38:03 2016	798
Thu Oct 13 08:38:35 2016	2120
Thu Oct 13 08:39:05 2016	793
Thu Oct 13 08:39:37 2016	0
Thu Oct 13 08:41:18 2016	672
Thu Oct 13 08:41:51 2016	814

Figure 5: Example of OBU-RSU connection records

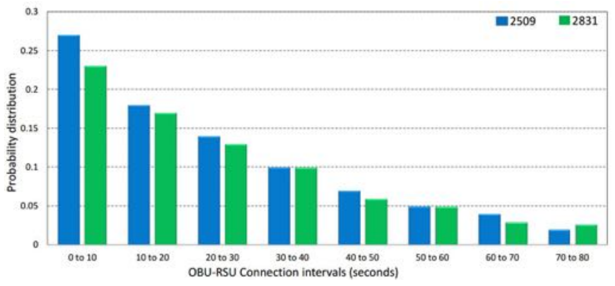


Figure 6: Probability distribution of OBU-RSU connections

records) and from four vehicles with bad quality of trace data (e.g., with 200000-220000 instances of SQL records). For each vehicle, we separate the available data into two parts: a learning part (L), and a testing part (T). The learning dataset L, which is the first 70% of a vehicle's trace data, is utilized to derive the Markov chain states and to calculate its transition probability matrix. The testing dataset T contains the rest 30% of trace data, which is used to test and evaluate the proposed algorithm. For instance if the duration of trace data is 10 months, which includes the trace files of 40 Saturdays, we use the data of the first 28 Saturdays for the learning purpose and apply the data from the remaining 12 Saturdays in the testing phase.

In order to feed the prediction algorithm with useful inputs, the original dataset has to be cleaned and processed into a proper format. As described in Subsection III-B, the raw data collected from the VANET testbed includes a huge amount of information. However, for the purpose of vehicle mobility prediction, the algorithm only needs the history of vehicles' connected RSUs and the timestamps of the connections. Therefore, we have to generate a file with all the connected RSUs/timestamps information for every vehicle, such that the vehicles' transition probability matrices can be calculated from it. In Figure 5, we show a small part of OBU-RSU connection records and the corresponding time stamps collected from the real-world VANET testbeds. The first five fields indicate the timestamps of the connections, and the last field indicates the connected RSU ID. Figure 6 shows the probability distribution of the OBU-RSU connection intervals for two vehicles (OBU IDs 2509 and 2797) randomly selected from the dataset. As we can see, 60% of the OBU-RSU connection has a duration of less than 30 seconds, and 80% of the OBU-RSU connection is less than 50 seconds.

IV. Implementation and evaluation

This section describes the implementation details, which include a real-world prototype using a testbed at the University of Aveiro (referred to as *Aveiro testbed*) and an implementation for large-scale evaluation an OpenStack infrastructure at the University of Bern (referred to as *Bern OpenStack testbed*). We use the Raspberry Pi platform to emulate the real communication between OBU-RSU on the *Aveiro testbed* consisting of 3 Raspberry Pi nodes for RSUs and mobile clients for OBUs. To validate the system scalability, we simulate a large-scale scenario with 40 RSUs on the *Bern testbed*. We use video streaming applications to evaluate the system performance, including vehicle prediction accuracy, network performance metrics, and metrics from the end-users' perspectives.

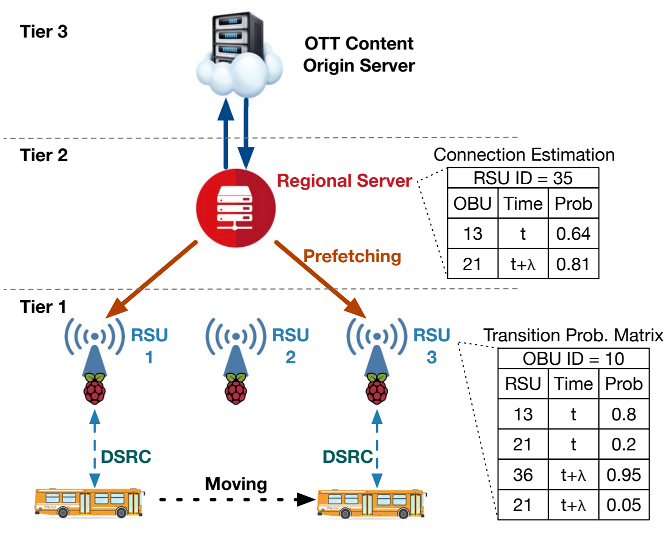
A. Prototype Implementation at Aveiro Testbed

In order to emulate the data transmission between an OBU-RSU pair in a practical way, we use Raspberry Pi nodes to implement all the RSU functionalities. For this scenario, three RSUs are emulated with three Raspberry Pi devices to implement the displacement of consumers along a route (as shown in Figure 7). The OBU starts from the connection with RSU 1, and it moves to RSU 2 or 3 according to the real vehicular mobility trace explained in Section III-B. Each emulated RSU represents a real RSU placed in the city of Porto, which the real IDs were assigned for each Raspberry Pi. The real IDs were chosen accordingly from the top three ranked IDs that received more contacts from the OBU ID 2509 during the experiment time window, e.g., RSU 1 ID=494, RSU 2 ID=2394 and RSU 3 ID=418.

We choose Raspberry Pi 2 Model B to emulate a RSU node. For all Raspberry Pi nodes, the OpenWRT operating system is installed. The client PC emulating the OBU consumer is a core I7 PC with 16GB RAM, 1T storage and 128 SSD, and with Ubuntu 16.04 installed. To build a HTTP and reverse proxy server, we choose Nginx (version 1.10) [24] in this experiment.

To emulate the vehicle mobility and its impact on OBU-RSU connections, we have used nmcli[1], a Unix-based network manager tool that reports the wireless SSID network status, to control the connectivity between RSUs and vehicles to emulate vehicle mobility. When an OBU needs to move from one connected RSU to another, the nmcli module manages the disconnection from the current RSU, and sets up the connection with the new RSU. To achieve this, we built a

[1]nmcli: https://developer.gnome.org/NetworkManager/stable/nmcli.html



Connection Estimation

RSU ID = 35		
OBU	Time	Prob
13	t	0.64
21	t+λ	0.81

Transition Prob. Matrix

OBU ID = 10		
RSU	Time	Prob
13	t	0.8
21	t	0.2
36	t+λ	0.95
21	t+λ	0.05

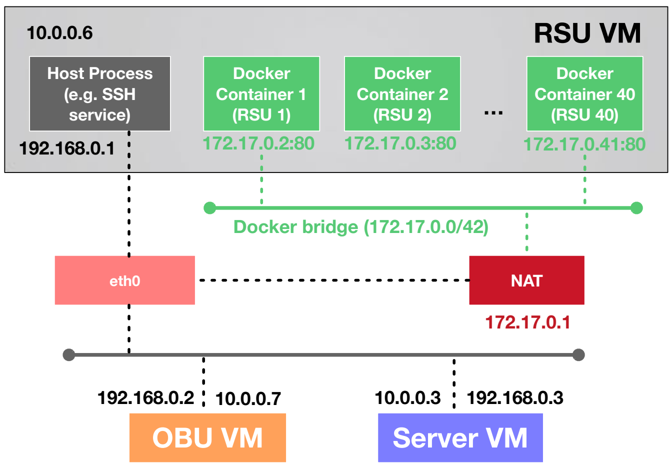
Figure 7: Prototype Implementation Topology at *Aveiro testbed*

RSU transition probability matrix (TPM), which decides the probability of moving from one RSU to another. This matrix is calculated using the vehicle mobility prediction algorithm presented in Section III-D. The TPM will be stored at every RSU, such that once a content request is received from a connected OBU, the RSU could decide which RSU will be the next connected one, which will trigger the nmcli network operations to break the old connection and establish the new connection with the RSU as indicated in the TPM.

B. Large-scale VANET Emulation at Bern OpenStack Testbed

To validate the performance of the proposed system in a large-scale testbed, we choose to use the OpenStack cloud infrastructure at the Institute of Computer Science at University of Bern to emulate a large VANET with multiple RSU nodes. Figure 8 shows the network topology, which includes 3 OpenStack VMs: one RSU VM includes 40 Docker containers and each container is running the identical OpenWRT image installed on the Raspberry Pi device, a client VM which sends video streaming requests to the RSU VM, and a server VM working as the content origin providing all video chunks. Detailed configurations of each VM are given below:

- One OBU VM as client with 1 vCPU, 2 GB of RAM and 20 GB of disk space. The VM client is an Ubuntu machine, which runs the same consumer services and tools of the PC client used in the previous evaluation.
- One Server VM with 2 vCPUs, 4 GB of RAM and 40 GB of disk space used as the Origin server VM running Window Server 2012, with Microsoft Smooth Streaming (MSS) installed



10.0.0.6
RSU VM
Host Process (e.g. SSH service)
Docker Container 1 (RSU 1)
Docker Container 2 (RSU 2)
...
Docker Container 40 (RSU 40)
192.168.0.1
172.17.0.2:80
172.17.0.3:80
172.17.0.41:80
Docker bridge (172.17.0.0/42)
eth0
NAT
172.17.0.1
192.168.0.2
10.0.0.7
10.0.0.3
192.168.0.3
OBU VM
Server VM

Figure 8: OpenStack Experiment Topology at *Bern testbed*

over an Internet Information Services which supports IIS Media Services, supporting to request a full Manifest or a specific video content fragment with given quality level.

- One RSU VM with 2 vCPUs, 4 GB of RAM and 40 GB of disk space. The RSU VM is an Ubuntu machine, which hosts 40 Linux Docker containers as 40 RSUs. A YAML script is created on the RSU VM to automatically configure and deploy 40 RSU containers. Each RSU communicates with the connected OBU using the same IP address provided by the Docker engine to accept incoming video requests. Different from the Raspberry Pi platform, which uses the nmcli tool to manage the OBU-RSU connections, we assume that all RSUs are directly connected to the OBU in the OpenStack experiments. Therefore, when an OBU moves to another RSU following its movement trace, the OBU-RSU connection will be established automatically. In this case, the prefetching mechanism only needs to point out the Ethernet IP address of the next targeted RSU, and conducts the prefetching operations.

C. *Evaluation Setup*

Regarding the setup of the prefetching evaluation, we apply nearly 1700 content object requests (one hour of observation time) for the prototype experiments on the *Aveiro testbed* and nearly 9400 content object requests (five hours of observation time) for OpenStack-based experiments on the *Bern testbed*. To validate the performance of the mobility prediction algorithm and its impact on the proposed mobility prediction-based prefetching, we compare our system (referred to as *Prediction*) with an optimal solution (referred to as *Reference*). The *Reference* solution uses the whole vehicle movement trace to make the prefetching operations, which means an OBU knows exactly which RSU will be the next connected RSU. Detailed descriptions of the two approaches are given as follows:

1) Reference: This approach uses the real VANET traces for an OBU_ID (e.g., 2509) and its input parameters are OBU_ID, RSU_ID and the amount of time that the OBU will be connected to the RSUs (as shown in Figure 5). In the *Reference* approach, the OBU will follow all the OBU-RSU connection information available in the data trace. This means that the future connected RSU and the connection times will be known beforehand such that the prefetching operations can be made with 100% accuracy about which RSU will be the next connected one.

2) Prediction: This approach uses the information of the predicted OBU's future connected RSUs to feed the proposed prefetching mechanism to have more accurate decisions on where the OTT content should be prefetched. The *Prediction* approach is based on vehicles' historical movement trajectories, and its main parameters are described in the following:

- OBU_ID is the ID of the OBU considered in the evaluation.
- Current_RSU_ID is the ID of the current connected RSU. This field is very important for forecasting. It is only considered as a valid prediction if the Current_RSU_ID hits the RSU_ID at that time instant.
- Next_RSU_ID is the next RSU, where the OBU can connect in the near future.
- TPM presents transition probability matrix of an OBU_ID when a certain probability is greater than 60%. Higher probabilities are also used as a tie-breaker in order to choose the ideal RSU candidates to receive the prefetched content.
- Granularity is synchronized accordingly with the experimentation time. It is about how often the prefetching mechanism should be updated.

When an OBU_ID starts the experiment (using the *Reference* approach as an example), an internal timer is triggered. This timer is used to check the TPM matrix if there are some probabilities to be connected with the next RSU_ID. For the experiments on the *Aveiro testbed*, Raspberry Pi devices are configured with a cache size of 500 MB. Five rounds of experiments are conducted with a confidence interval of 95%. For the OTT content requests, we use a real application asking for a chunk every 2 seconds (configured as default in the consumer's manifest) of a certain bitrate/quality. When the video ends, but the experiment is still running, the consumer starts the video streaming again until the experimentation is finished. The whole experiment is considered for 1 hour. OBU movements are emulated using the wireless interface of the PC, while prefetching requests are requested to the respective RSU using the backend

Table II: Available Qualities and Average Size per Chunk

Available Qualities [bps]	Average Chunk Size [bytes]	Available Bitrate [bps]	Average Chunk Size [bytes]
230,000	58,620	991,000	240,559
331,000	82,529	1,427,000	347,243
477,000	117,652	2,056,000	499,551
688,000	169,255	2,962,000	718,157

Ethernet network. For client HTTP request management purposes, we used the urllib3[2] tool, which is an abstraction for a collection of ConnectionPools used to support HTTP requests to multiple hosts. Each prefetching request uses a different thread such that it is possible to have multiple in parallel HTTP requests. To evaluate the Quality of Experience (QoE), a QoE probe is also installed in the client, which evaluates the QoE on watching adaptive streaming videos (more technical details about the QoE probe can be found in [25]). The QoE probe basically estimates the consumer experience as well as helps to understand the network infrastructure performance and limitations (e.g., influence of jitter, delay, link quality from the RSU up to the origin servers).

For the OpenStack experiments on the *Bern testbed*, most of the parameters are identical with the ones used in the *Aveiro testbed*. The only difference is that the experiment time window is randomly chosen over five hours from Wednesdays (the highest accuracy day) for OBU_ID 2509 with good quality. Both the *Reference* and *Prediction* approaches consider different Wednesdays to avoid biased predictions for the same day.

D. *Performance Metrics*

For all experimental evaluations, the video streaming considered is Elephants Dream H.264 720p[3]. Table II shows the available of video qualities and the average chunk sizes. To evaluate the system performance, we consider the following metrics:

Average Load measures the mean of the number of processes waiting to be served by the CPU. Assuming a single-CPU system, the critical load point is 1. An average load value of 0 means that there are no processes waiting to be served by the CPU, and average load value is 1 means that the processor is at full capacity. In case of multi-processors, the critical load point is given by the number of core processors available.

[2]https://pypi.python.org/pypi/urllib3

[3]https://www.microsoft.com/en-us/download/details.aspx?id=18199

Backend traffic is a key metric in content delivery networks. As the purpose of RSU caches is to reduce the load on backend servers measured by the percentage of misses that reaches the origin servers, a low metric indicates good performance.

Vehicle Mobility Prediction Accuracy measures the accuracy of the location prediction algorithm. We randomly select 10% of states out of the Markov chain states (e.g., states from 9 AM to 11 AM) derived for each particular weekday from the training data set L for each vehicle. Afterwards, the prediction algorithm is performed for each of the selected states to estimate the possible future connected RSU(s) in the next λ minutes. These states have been chosen as the random testing points. We check the transition probability for states during the same period of time in the testing data set T as well. Afterwards, the Mean Absolute Error (MAE) of the possible transitions of the corresponding testing points is calculated according to Equation 11.

$$MAE = \frac{1}{M}\sum_{i=1}^{M} \mid Pr_i L - Pr_i T \mid \quad \& \quad \mathit{Accuracy} = (1 - MAE) \times 100 \tag{11}$$

Cache Hit-Ratio summarizes the ratio between the number of cache hits when compared to the number of cache requests, and is an indicator of how good the caching algorithm is on guessing programs that will be requested in the near future. The miss percentage might be used to calculate the backend traffic from, for example, the RSU to the origin server. The calculation of the hit ratio and miss ratio are given in Equations 12.

$$\mathit{Ratio}_{Hit} = \frac{Total_{Hit}}{Total_{Hit} + Total_{Miss}} \quad \& \quad \mathit{Ratio}_{Miss} = \frac{Total_{Miss}}{Total_{Hit} + Total_{Miss}} \tag{12}$$

Request Latency represents the time required to serve a request. Latency must be carefully monitored to ensure that there is no impact on the user experience, as high request latency may indicate high server or network load, which leads to queued or dropped requests.

Mean Opinion Score (MOS) is a common QoE measure. Due to its subjective nature, QoE evaluations vary significantly between different users. However, objective QoE evaluation frameworks exist that provide an estimate on the expected MOS of a given service. A developed Smooth Streaming QoE estimation probe [25] is used to provide the objective MOS estimate.

V. Experimentation Results

In this section, we present and discuss the results of both the prototype implementation on the *Aveiro testbed* and the OpenStack experiments at the *Bern testbed*.

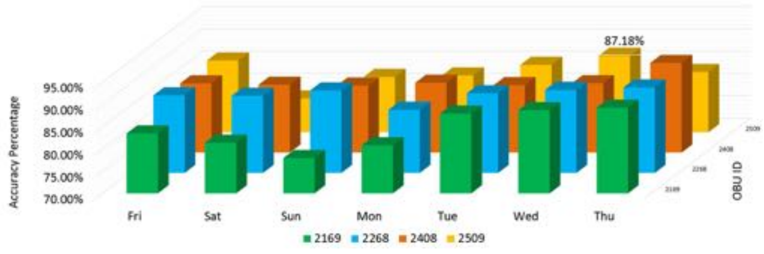


Figure 9: Prediction accuracy of *hybrid predictor* for OBUs with good quality

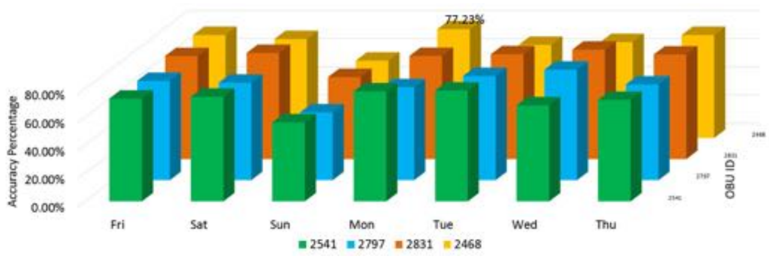


Figure 10: Prediction accuracy of *hybrid predictor* for OBUs with poor quality

A. Mobility Prediction Accuracy Results

Figures 9 and 10 show the accuracy of the proposed *hybrid predictor* per day, for users with good and poor quality of trace data. As described before, we classify vehicle trace quality based on the number of SQL entries of a vehicle during the data collection period. We randomly choose 4 OBU_ID (2169, 2268, 2408, 2509) from the group of *good quality trace user*, and another 4 OBU_ID (2541, 2797, 2831, 2468) from the group of *poor quality trace user*.

As we can see from Figure 9, the *hybrid predictor* can reach a prediction accuracy of nearly 88% for OBU_ID 2509 on Wednesday, and the average accuracy of OBU_ID 2509 over all the weekdays and weekends is 84%. The average accuracy of all the four OBUs with good quality data traces is nearly 86%. Even for the OBUs with poor quality data traces, we can see from Figure 10 that the *hybrid predictor* can reach an accuracy of 77% for OBU_ID 2468 on Monday, and the average accuracy of OBU_ID 2468 over all the weekdays and weekends is around 69%. The average accuracy of all the four OBUs with poor quality data traces is nearly 70%. These results show that the *hybrid predictor* is able to generate accurate enough prediction for users with both good and poor trace qualities. Given that this work focuses on OBUs that are installed on buses that are supposed to follow predefined fixed driving routes an average accuracy of 86% for buses with good trace quality is still not as high as we could expect. This is because during the 10 months of data collection, the same bus (with the same OBU_ID) might be assigned to different routes, which brings uncertainty for prediction.

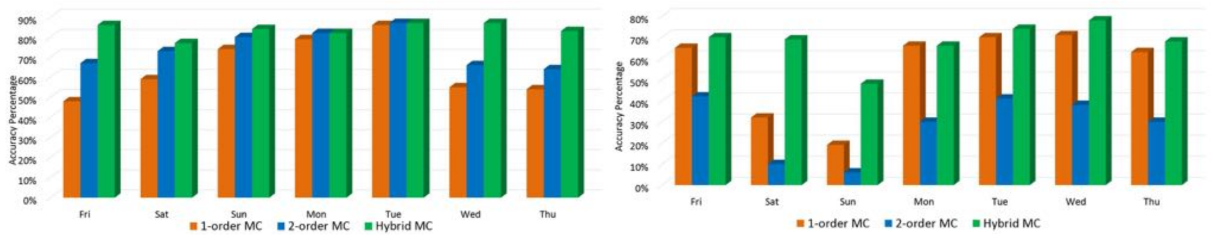


Figure 11: Prediction accuracy of *hybrid predictor* for OBU_ID 2509 (left) and 2797(right)

To clearly show the benefits of the *hybrid predictor*, we compare the prediction accuracy of the first-order Markov chain, the second-order Markov chain, and the *hybrid predictor*. From Figures 9 and 10, we randomly select OBU_ID 2509 and OBU_ID 2797 as the representatives of the good and bad quality data category. Figures 11 shows the prediction accuracies of the first-order Markov chain, the second-order Markov chain, and the proposed *hybrid predictor* for OBU_ID 2509 and OBU_ID 2797. As we can see, the *hybrid predictor* outperforms the first-order and the second-order Markov chains for both good and poor quality datasets. However, there is a significant performance difference between the first-order and the second-order Markov chains when the traces are of different quality. As shown in Figure 11 (left), when vehicles' movement data is of good quality (OBU_ID 2509), the second-order model could reach an average accuracy of 75%, which is much better than the first-order model (an average accuracy of 65%). This result matches with our statement about the performance of the second-order Markov chain described in Section III-D. But even for good quality movement data, there are still occasions that the OBU-RSU connection data is not recorded due to data transmission collision or other technical problems of the VANET testbeds. Therefore, the *hybrid predictor* can adaptively switch to the first-order mode in those cases when it detects that only one order data is available, which leads to a better performance than the second-order Markov chain. When vehicle's movement data is of poor quality, as shown in Figure 11 (right) for OBU_ID 2797, the assumptions of the second-order Markov chain's superior performance do not hold anymore. Therefore, it can not make more accurate predictions than the first-order Markov model. However, the proposed *hybrid predictor* will apply the first-order Markov model in this condition, and whenever there are some 2-state information available, it will switch to the second-order Markov model, which enhances the performance of the first-order Markov chain. That is why the *hybrid predictor* performs better than the first-order Markov model.

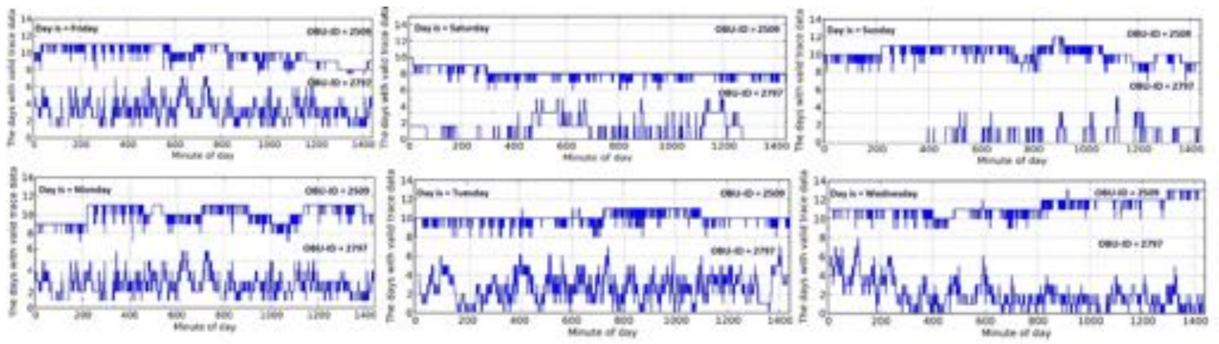


Figure 12: Trace qualities of OBU_ID 2509/2797 on weekdays.

As shown in Figures 9 and 10, the *hybrid predictor* can achieve an average prediction accuracy of 83% for OBU_ID 2509 (with good quality trace data), and 68% for OBU_ID 2797 (with bad quality trace data). To explain the performance difference of these two vehicles, we next discuss the data quality of OBU_ID 2509 and 2797. Figure 12 presents the number of valid data records stored in the SQL table for OBU_ID 2509 and 2797 during a time period of 24 hours (1440 minutes) of all weekdays. The metric of *the days of valid trace data* counts the number of days that have more than a certain number of records, referred to as *threshold*, stored in the SQL database for one day. Take Monday as an example: A value of 12 means that there are 12 Mondays that have more than the *threshold* number of records at a specific time moment stored. For instance, a *threshold* value of 1000 means that if an OBU_ID has less than 1000 records in one day, the data of that specific day will not be included in the prediction. This is because such a low number of records happen most probably for vehicles that are not in the regular operational function and could be under the conditions of vehicle breakage or maintenance. Therefore, their data are exceptional cases and should not be considered in the prediction procedure. In Figure 12, a pulse means that at the corresponding moment of time, there are certain weekdays with valid trace data. Therefore, a good quality trace data should include a series of continuous pulses at a high value of *the days of valid trace data*. Instead, a series of scattered pulses with large fluctuations and a low value of *the days of valid trace data* means that the data is of very bad quality.

We can see that OBU_ID 2509 has more valid trace data recorded than OBU_ID 2797 for all the weekdays, which leads to better performance. Moreover, we see that for OBU_ID 2797, there are pulse series with large fluctuations and very low values of *the days of valid trace data*

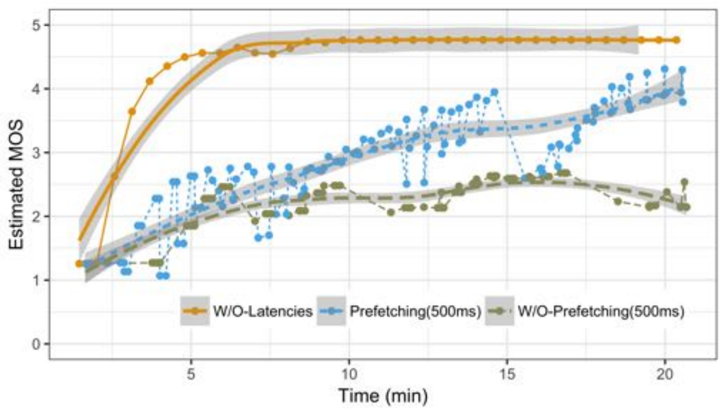


Figure 13: Mean Opinion Score (MOS) Results of Experiments on the Aveiro Testbed

for every weekdays. Figure 11 (right) shows that for OBU_ID 2797, the *hybrid predictor* can only achieve an accuracy of around 50% for Sunday. This can be further explained by checking the trace quality of OBU_ID 2797 on Sunday, which has a long period of no valid trace data (the first data recording starts from around 06:30 AM). For OBU_ID 2509, it has a series of continuous pulses at a high value of *the days of valid trace data*. For example, on Tuesday from 11:00 AM to 18:30 PM or Wednesday from 11:00 AM to 16:00 PM, it has continuous pulses with the *the days of valid trace data* value equal to 11 or 12. This leads to a relatively high prediction accuracy for OBU_ID 2509 on Tuesday and Wednesday (as depicted in Figure 11). In contrast, when OBU_ID 2509 has continuous pulses at a medium value of *the days of valid trace data*, its performance will drop. For instance, on Saturday from 05:00 AM to 24:00 PM, it has continuous pulses with low value of *the days of valid trace data* equals to 8. That is why we see a relatively low prediction accuracy (78%) for OBU_ID 2509 on Saturday with *hybrid predictor* (as depicted in Figure 11).

B. Prototype Evaluation Results on Aveiro Testbed

This section presents evaluation results of the proposed prefetching mechanism on the prototype implementation at the *Aveiro testbed* (as shown in Figure 7) with real wireless connections between OBUs and Raspberry Pi nodes.

Figure 13 presents the estimated MOS of the OBU consumer when it is watching an OTT video content. For this evaluation, the entire video manifest including eight default qualities (230000 bps, 331000 bps, 477000 bps, 688000 bps, 991000 bps, 1427000 bps, 2056000 bps, 2962000 bps) were settled to be consumed by the QoE probe. The QoE probe emulates the OBU consumer and it has a dynamic behaviour, adapting itself (automatically switching between qualities) when the backend link conditions start to get poor, e.g., high jitter and latency, poor bandwidth, etc. To

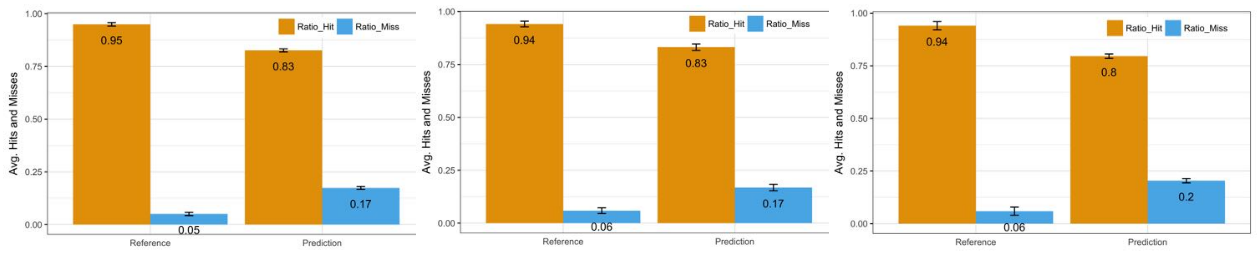


Figure 14: RSU Cache Hit/Miss Ratios of Experiments on the Aveiro Testbed: with Low (a), Medium (b) and High (c) Bit Rates

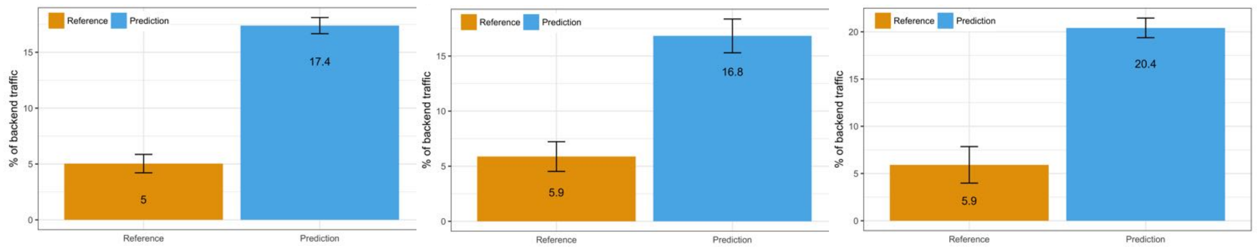


Figure 15: RSU Total Backend Traffic of Experiments on the Aveiro Testbed: with Low (a), Medium (b) and High (c) Bit Rates

test the impacts of poor backend traffic conditions on OBU consumer's QoE, we have introduced an intermittent latency of 500ms on the link between the RSUs and the OTT origin server. As it can be seen in Figure 13, the testbed infrastructure as it is (w/o latencies), means without any injected latency between the RSUs and the Origin server, which leads to better MOS values, remaining stable until the end of this experiment. Therefore, we analysed the MOS of the OBU consumer with injected latency, and with and without applying the proposed prefetching mechanism. With prefetching mechanism, a better performance is observed (even with MOS fluctuations) compared to without prefetching mechanism. The QoE probe is constantly trying to decrease video qualities influenced mostly by the link latency, which explains the perceived MOS fluctuations. Therefore, this QoE analysis reinforces that the prefetching mechanism leads to better MOS improvements even facing poor backend communication links, which is one of the main factors degrading the MOS.

Next, we evaluate the impact of content hit and miss rates. The results are presented in Figure 14. We have considered three different fixed bit rates (low, medium and high). Starting from the lowest bit rate (as shown in Figure 14 (a)), it is possible to observe that the optimal *Reference* always provides the best performance, while the *Prediction* approach closely follows the optimal approach with 12% less hits. Followed by this first analysis, the medium and high video quality

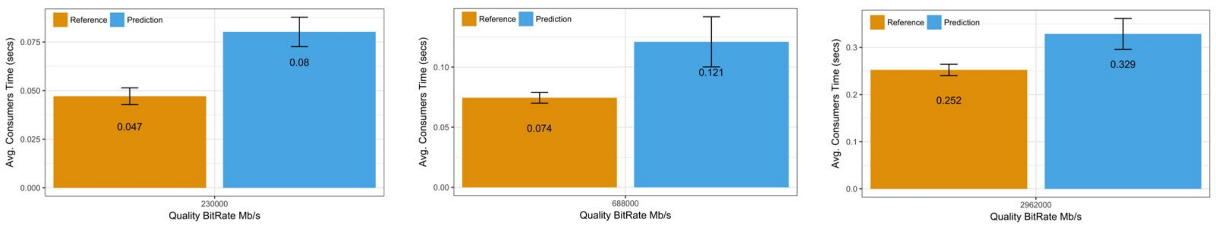


Figure 16: Average Consumers Time of Experiments on the Aveiro Testbed: with Low (a), Medium (b) and High (c) Bit Rates

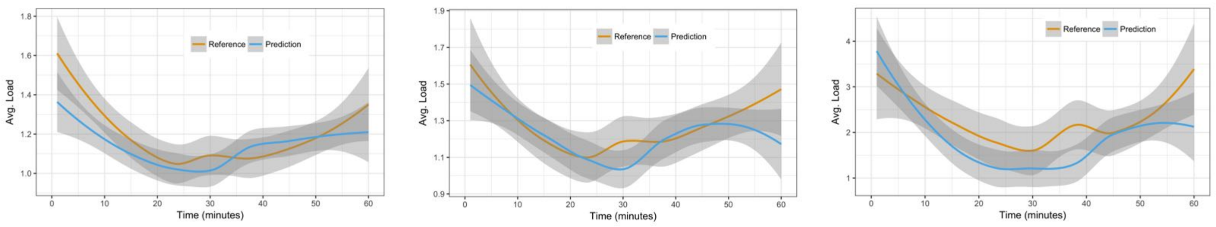


Figure 17: Average RSU CPU/Load/Memory Consumption of Experiments on the Aveiro Testbed: with Low (a), Medium (b) and High (c) Bit Rates

bit rates present also very small differences. Thus, we could conclude that the *Prediction* approach can save caching operations and costs by placing the correct future OTT content chunks on the right RSUs along the vehicle mobility path. The ideal *Reference* prefetching approach provides the best overall caching performance, which is kept approximately with a 95% hit-ratio instead a 100%. This approximately 5% misses probably as caused by the OBU-RSU connection and disconnection processes, which add some extra delays.

The volume of backend data transfers is a crucial QoS metric that determines the amount of data transfer between the RSU and backend origin server. Efficient prefetching will significantly reduce the frequency of RSU-backend server data transmission, which will lower costly bandwidth consumption. In general, an accurate prediction of an OBU's future connected RSU will enable the content to be ready and thus reduce the OTT content being requested from the OTT content origin. As we can see, taking the case of high bit rate consumption (as shown in Figure 15 (c)) as an example, the differences between the *Reference* and the *Prediction* approaches are 14% approximately, 10% for medium quality and 12% for the lowest quality. Therefore, regardless of quality differences, the *Prediction* approach with an accurate TPM matrix is able to play an important role on reducing the total backend traffic between RSUs and OTT origin server.

Figure 16 shows the average OBU request time for each video quality. At a first glance the *Prediction* slightly consumes more time, but the delay is very small compared to the *Reference*

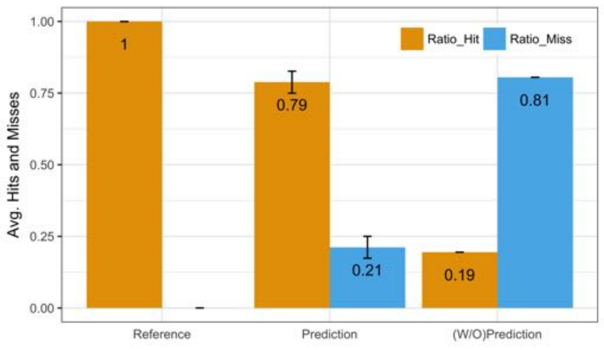


Figure 18: RSU Cache Hit-Miss Ratio of Experiments on the Bern Testbed

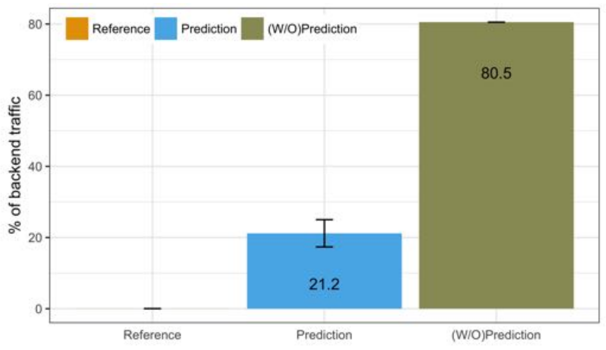


Figure 19: RSU Total Backend Traffic of Experiments on the Bern Testbed

approach over different bit rates (low: 0.033s, medium: 0.047s and high: 0,077s). By taking the optimum *Reference* approach as a base, it is also possible to conclude that the *Prediction* approach has an acceptable response time considering that the OBUs are wirelessly connected to the RSUs. Moreover, the differences are not big, showing that even though the prediction algorithm might not always predict the correct future connected RSU ID, the *Prediction* approach has a very stable performance over different bit rates on average. It is also possible to verify that the chunk size (e.g., high quality) has also some influences on backend traffic, which means more data will flow through the network. In the case of scenarios with real wireless connections, the unstable wireless connections can be a non-ignorable factor for the long average request time.

Figure 17 shows the average CPU load of those respectively emulated RSUs for both the *Reference* and the *Prediction* approaches. We can see that there are no extreme high load consumption of the RSUs. Note that the RSUs, which are emulated by the Raspberry Pi nodes, have limited available resources, and even with those hardware limitations, the prediction approach could be executed normally without consuming additional high processing time and system resources.

C. OpenStack Experiment Results on Bern Testbed

This subsection presents evaluation results of the proposed prefetching mechanism when deployed in a large-scale scenario with 40 RSU emulated in an OpenStack infrastructure at the *Bern testbed*. The network topology of the experiment can be found in Figure 8. Considering the high bit rate of 2962000 bps as the chosen quality for all evaluations presented in this subsection, Figure 18 presents the hit-ratio performance of the *Reference*, *Prediction* and *(W/O) Prediction* approaches, where the *Prediction* approach enables significant cache cost savings. The *Prediction*

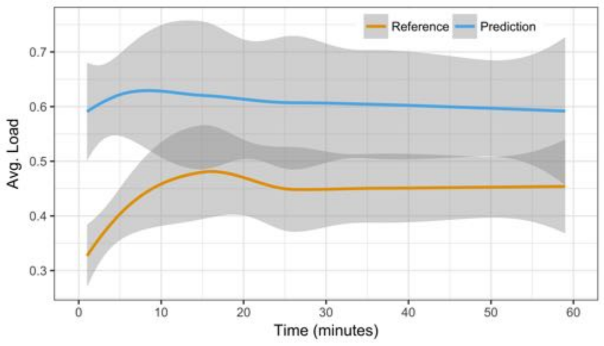


Figure 20: RSU CPU Loads of of Experiments on the Bern Testbed

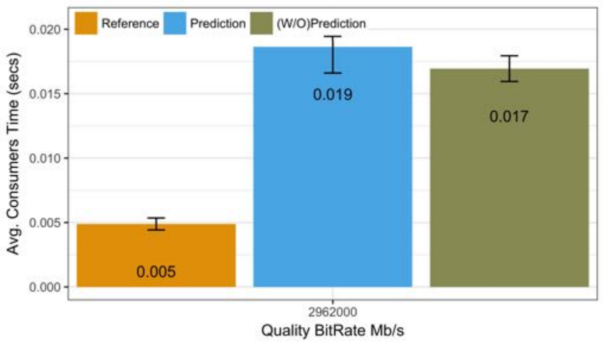


Figure 21: Average Consumer Times of Experiments on the Bern Testbed

approach has 21% less hits ratios compared to the *Reference*, and 60% more hits ratios compared to the *(W/O) Prediction* approach. So, its performance and accuracy are acceptable, since in this scenario the *Prediction* approach has more RSUs options during the TPM matrix processing procedure. By using the *Prediction* approach, there are also significant RSU cache cost savings and also a positive impact on bandwidth savings over urban medium-scale environments.

Figure 19 shows that the total backend traffic in the *Prediction* approach is bigger than that for the *Reference* approach. This is similar to the *Aveiro testbed* case. This growth is again because more RSUs are deployed in the network, which are demanding more frequent RSU TPM decisions (every 60 seconds or even shorter). In these situations, any wrong decision about the TPM will increase RSU-backend OTT server data transmissions. Finally, compared to *(W/O) Prediction* approach there are approximately 60% less data requests transmission to the backend OTT server, which is satisfactory considering the amount of decisions that needs to be performed during the prediction of the OBU's movements.

Figure 20 presents the CPU load and memory usage of all RSUs during the prefetching operations. In the *Prediction* approach the CPU load is higher than that for the *Reference* approach. However, considering the number of cores of the emulated RSUs, this behaviour is reasonable and acceptable. For example, the load cannot reach more than 4 for a RSU with 4 CPU cores. Compared to the *Aveiro testbed* results, even the CPU load consumption is higher, the overall RSU performance is not degraded in terms of user request latency. This difference probably is because when prefetching needs to take some decisions about where will be the next RSU, it has to check first the TPM matrix to take a reasonable decision. This explanation is reinforced by the *(W/O) Prediction* approach results, which do not use any decision calculations, remaining in average, below than both approaches of *Reference* and *Prediction*.

Figure 21 shows the average OBU request time in the *Bern testbed* experiments. The difference between the approaches are very small (0.014 seconds compared to the *Reference* and 0.02 seconds with *W/O Prediction*). The very stable Ethernet (1Gb/second) connection between all OBU-RSU-Servers leads to better OBU consumer time performance compared to the *Aveiro testbed* presented in the Figure 16. The *Aveiro testbed* includes the real OBU-RSU wireless WAVE connections, which are not as stable as in the Bern OpenStack testbed.

VI. Conclusions

Multimedia content delivery in VANETs is challenging due to node movements. This paper proposed a vehicle mobility prediction-assisted OTT content prefetching mechanism in hierarchical VANETs. We designed a novel approach combining highly accurate forecasts of the OBU's mobility with a lightweight OTT content prefetching mechanism. This combination enables us to leverage accurate vehicle mobility prediction to provide outstanding system performance for VANET OTT content prefetching. The system performance has been validated using a rich dataset collected from a large-scale real-world VANET testbed. Experimental results show that the proposed approach provides good OTT content delivery QoE results and it enables significant RSUs cache and bandwidth savings, which bring benefits for both end-users and OTT service providers. Future improvements will focus on adding real-time prediction and exploring the prefetching approach when facing with unknown content.

References

[1] X. Cheng and et.al, "5g-enabled cooperative intelligent vehicular (5genciv) framework: When benz meets marconi," *IEEE Intelligent Systems*, vol. 32, no. 3, pp. 53–59, May 2017.

[2] S. Momeni and B. E. Wolfinger, "Availability evaluations for iptv in vanets with different types of access networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, p. 117, Jul 2014. [Online]. Available: https://doi.org/10.1186/1687-1499-2014-117

[3] C. Quadros and et.al, "Beacon-less video streaming management for vanets based on qoe and link-quality," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015.

[4] R. Kim, H. Lim, and B. Krishnamachari, "Prefetching-based data dissemination in vehicular cloud systems," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 1, pp. 292–306, Jan 2016.

[5] S. Eichler, "Performance evaluation of the ieee 802.11p wave communication standard," in *2007 IEEE 66th Vehicular Technology Conference*, Sept 2007, pp. 2199–2203.

[6] C. Sommer, R. German, and F. Dressler, “Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis,” *IEEE Transactions on Mobile Computing*, pp. 3–15, January 2011.

[7] M. Aramrattana and et. al, “A simulation framework for cooperative intelligent transport systems testing and evaluation,” *Transportation Research Part F: Traffic Psychology and Behaviour*, 2017.

[8] M. Cesana and et.al, “C-vet the ucla campus vehicular testbed: Integration of vanet and mesh networks,” in *2010 European Wireless Conference (EW)*, April 2010, pp. 689–695.

[9] C. Ameixieira and et.al, “Harbornet: a real-world testbed for vehicular networks,” *IEEE Communications Magazine*, pp. 108–114, September 2014.

[10] J. Harri, M. Fiore, F. Filali, and C. Bonnet, “Vehicular mobility simulation with vanetmobisim,” *SIMULATION*, pp. 275–300, 2011.

[11] A. S. Gomes, B. Sousa, D. Palma, V. Fonseca, Z. Zhao, E. Monteiro, T. Braun, P. Simoes, and L. Cordeiro, “Edge caching with mobility prediction in virtualized lte mobile networks,” *Future Generation Computer Systems*, pp. 148 – 162, 2017.

[12] J. Famaey, F. Iterbeke, T. Wauters, and F. D. Turck, “Towards a predictive cache replacement strategy for multimedia content,” *Journal of Network and Computer Applications*, pp. 219 – 227, 2013.

[13] G. Nencioni and et.al, “Understanding and decreasing the network footprint of catch-up tv,” in *Proceedings of the 22Nd International Conference on World Wide Web*, ser. WWW ’13. New York, NY, USA: ACM, 2013, pp. 965–976.

[14] Z. Su and et.al, “The next generation vehicular networks: A content-centric framework,” *IEEE Wireless Communications*, vol. 24, no. 1, pp. 60–66, February 2017.

[15] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, “A parsimonious model of mobile partitioned networks with clustering,” in *2009 First International Communication Systems and Networks and Workshops*, Jan 2009, pp. 1–10.

[16] N. Abani, T. Braun, and M. Gerla, “Proactive caching with mobility prediction under uncertainty in information-centric networks,” in *Proceedings of the 4th ACM Conference on Information-Centric Networking*. ACM, 2017, pp. 88–97.

[17] F. Zhang and et.al, “Edgebuffer: Caching and prefetching content at the edge in the mobilityfirst future internet architecture,” in *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2015, pp. 1–9.

[18] G. Mauri, M. Gerla, F. Bruno, M. Cesana, and G. Verticale, “Optimal content prefetching in ndn vehicle-to-infrastructure scenario,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 3, pp. 2513–2525, March 2017.

[19] R. Monteiro, L. Guedes, T. Condeixa, F. Neves, S. Sargento, L. Guardalben, and P. Steenkiste, “Lessons learned from a real vehicular network deployment of delay-tolerant networking,” in *IEEE ICCW workshop*, June 2015, pp. 2489–2494.

[20] J. Silva, A. Dias, J. Nogueira, L. Guardalben, and S. Sargento, “Content-aware prefetching in over-the-top wireless networks,” in *IEEE Symposium on Computers and Communications (ISCC)*, July 2017.

[21] M. Karimzadeh and et.al, *Mobility and Bandwidth Prediction as a Service in Virtualized LTE Systems*. IEEE Cloudnet, 10 2015, pp. 132–138, 10.1109/CloudNet.2015.7335295.

[22] A. Zolnierek, *The Empirical Study of the Naive Bayes Classifier in the Case of Markov Chain Recognition Task*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 329–336.

[23] B. Sousa. and et.al, “Enabling a mobility prediction-aware follow-me cloud model,” in *IEEE LCN*, December 2016.

[24] “NGINX - HTTP server and reverse proxy,” accessed: 2017-11-25. [Online]. Available: https://www.nginx.com/

[25] A. Salvador and et.al, *QoE Assessment of HTTP Adaptive Video Streaming*. Cham: Springer International Publishing, 2015, pp. 235–242.