RapidVFetch: Rapid Downloading of Named Data via Distributed V2V Communication

Zhiyi Zhang
UCLA
zhiyi@cs.ucla.edu

Tianxiang Li
UCLA
tianxiang@cs.ucla.edu

John Dellaverson
UCLA
jdellaverson@cs.ucla.edu

Lixia Zhang
UCLA
lixia@cs.ucla.edu

***Abstract*—As the world becomes increasingly online, vehicular networking is becoming a reality. Vehicular networks require asynchronous, secure, and high-performance data transfers. TCP/IP networking, which is designed for infrastructure-based and synchronous communication, does not match vehicular networks well. Therefore, there has been an increased recognition that vehicular networking can be better served by a data-centric networking model.**

This paper contributes to this new recognition by proposing RapidVFetch, an infrastructure-to-vehicle data downloading protocol based on Named Data Networking (NDN). RapidVFetch leverages V2V communication over NDN for data requesters to initiate a prefetching process that does not require any centralized component for content preallocation and is transparent to the infrastructure. At the same time, RapidVFetch utilizes NDN's built-in data-centric security support to ensure the integrity, authenticity, and confidentiality of downloaded data. Our evaluation results from simulations show that, assuming available neighbor vehicles, RapidVFetch can greatly improve data downloading performance by prefetching data at desired locations and can provide network stability by fetching data directly from other vehicles.

***Index Terms*—Prefetching, Vehicular Networking, Named Data Networking**

I. Introduction

Vehicular technology has been steadily improving, with content distribution to vehicles being one of the major application use cases. Many applications such as multi-media, navigation, software updates, as well as real-time interactive applications have different demand for data downloading performance. Traditionally, data downloading to vehicles is achieved solely through Road Side Unit (RSU), using V2I or V2R communication. Because of the coverage limits of RSUs, data prefetching has been proposed as an approach to facilitate vehicular data downloading. The main idea is to preallocate content into RSUs that vehicles will drive through in the future so that vehicles can retrieve data without waiting for a long round-trip time to original content producers.

As observed, vehicular networking is a very different setting compared with traditional synchronous and channel-based networks and thus has different requirements on the communication model. (i) It requires asynchronous communication along with in-network storage to enable prefetching and cope with intermittent connectivity. (ii) Correspondingly, a channel-based security model is not sufficient. Instead, data-centric security fits better for data integrity, authenticity, and confidentiality in asynchronous communication. (iii) In addition, V2V, as an important component of vehicular networking, requires an infrastructure-free communication model to fully utilize short-lived wireless links among vehicles.

Designed for infrastructure-based communication, TCP/IP networking uses IP addresses to perform point-to-point packet delivery, which does not fit well into vehicular networking. (i) IP assumes a synchronous communication model through one-to-one channels. Moreover, stateless IP forwarding does not support in-network storage; higher layer services like content delivery networks (CDN) are designed for a much narrower scope of applications and incur additional costs. (ii) The TCP/IP network model itself does not provide security. Later patches like IPsec and TLS pose a channel-based security model, which isn't sufficient for highly mobile scenarios. (iii) In wireless V2V communication, IP addresses lose their meaning and there is no real "point-to-point" communication. Also, intermittent connectivity and mobility inevitably leads to extra complexity for addressing and channel tear-down/re-establishment.

On the other hand, there has been an increased recognition that vehicular networking can be better served by a data-centric network architecture [1] [2]. Named Data Networking (NDN) [3], as a proposed Internet architecture, enables a new connection-less communication model where applications fetch the desired Data packets via Interest packets carrying desired data names (Figure 1). In NDN: (i) Data fetching is asynchronous: data can be fetched from any in-network devices who have it. (ii) Data is secured in a data-centric manner: data is directly protected by the digital signature and/or content encryption, independent from where data is kept. (iii) Data retrieval is decoupled from channels identified by IP addresses. These key properties of NDN enable an efficient realization of a communication model for vehicular networks. In this paper, we will use the prefetching scenario as a specific use case to illustrate the advantages of NDN in vehicular network content distribution.

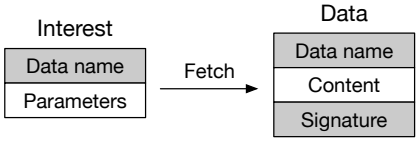


Figure 1: Interest and Data packets in NDN

So far, a number of works have been proposed to explore

978-1-7281-4571-6/19/$31.00 ©2019 IEEE

prefetching mechanisms in vehicular networking, including both IP based and NDN based solutions. The main direction is to employ a centralized service to predict a vehicle's future location and desired data. Based on the prediction result, RSUs prefetch and cache the data for future use. However, such mechanisms may make mistakes in location and content prediction, resulting in waste of both bandwidth and storage. In addition, RSUs have to be aware of their protocols so as to parse the downloaded content and the security of prefetched content is hard to guarantee.

Contribution In this paper, we propose RapidVFetch to facilitate vehicular data downloading by prefetching named, secured data via V2V communication over NDN. Intuitively, since it is the vehicle who best knows its future locations and desired content, RapidVFetch lets each vehicle, called a *requester*, express their needs by Interest packets which are small in size and solicit help from other vehicles, called *forwarding vehicles*, through V2V communication. Vehicles at future locations can simply express these small Interest packets to the RSUs, where the prefetched packets will be cached at the RSUs and will soon be used by the requester when it moves into the RSU's range. For real-time applications, the data can also be forwarded back to the requester.

RapidVFetch provides the following desirable features.

- Having each requester take care of their own needs, RapidVFetch provides accurate content prefetching at desirable RSUs without reliance on centralized services.
- By utilizing forwarding vehicles, RapidVFetch allows a requester to maintain stable data downloading for real-time applications.
- Enabled by NDN's built-in security, RapidVFetch ensures data authenticity and confidentiality regardless of the trustworthiness of other vehicles or RSUs involved in the downloading process.
- RapidVFetch is transparent to the infrastructure and RSUs, that is, RSUs and backbone routers do not need to be aware of RapidVFetch.

Our work illustrates the superiority of using a data-centric network architecture in vehicular networking: NDN enables RapidVFetch to let vehicles themselves pick what to cache in desired RSUs without reliance on a predictor; at the same time, the prefetched data is secured in a data-centric way and the whole process can be transparent to the infrastructure.

Outline. In the rest of the paper, we introduce the distinguishing features of NDN in Section II. We then move on to discuss the problem statement and related works in Section III. We then provide an overview of RapidVFetch design in Section IV, describe the protocol details in Section V, and provide an evaluation of RapidVFetch's performance in Section VI. Section VII gives a discussion on RapidVFetch and we conclude our work in Section VIII.

II. Named Data Networking

A. Basic Concepts of NDN

Named Data Networking (NDN) is a proposed content-centric Internet architecture. NDN features three key concepts that fundamentally distinguish it from the existing TCP/IP architecture: naming, stateful forwarding, and data-centric security.

Naming. In NDN, applications directly name and request content. This means that applications pull desired content from the network by expressing an interest in a data name, rather than building an IP connection with another network node. Therefore, applications do not need to know where the target data is, merely what it is called. Moreover, the application layer defines how data is named, allowing for full flexibility and rich semantics at the network layer.

Stateful Forwarding. To forward a Data packet back to the Interest sender, NDN employs a stateful forwarding plane. An NDN forwarder forwards Interest packets while at the same time keeping a table of these Interests; when a corresponding Data packet comes back, the forwarder looks up the corresponding record and forwards the Data packet to where the Interest came from. Importantly, the Data will also be cached in the forwarder to satisfy future Interest packets requesting the same piece of data.

Data-centric Security. NDN builds in security [4] by directly securing every Data packet. When Data packets are produced, the data producer appends a digital signature to the data name and content. Content can also be encrypted if required by the application's security concerns. This stands in sharp contrast to the channel-based security model in today's TLS where data is only protected in a channel between two end hosts. With NDN's data-centric security, Data packets can be kept anywhere and a data consumer can verify its provenance and integrity regardless of where it was fetched from.

B. NDN and Vehicular Networking

Vehicular networks, by their nature, involve mobility and intermittent connectivity. It is thus desirable to provide in-network storage and asynchronous communication. Vehicles remain in the range of a given RSU for only a short period of time, which heightens the importance of fully utilizing that time and getting quick responses. In-network storage enables responses to arrive more quickly by moving data closer to its consumer. Asynchronous communication provides necessary flexibility in the context of intermittent connectivity. Moreover, the network must provide data security in order to protect against eavesdroppers or malicious actors.

Using named, secured data addresses all of these issues. (i) Naming data decouples communication from specific interfaces and endpoints, which enables a vehicle to utilize any available interfaces and fetch data from others as soon as physical connectivity exists. (ii) Widespread use of caching into the network nodes (both on RSUs and on vehicles) is naturally supported by the NDN protocol, facilitating both content dissemination and asynchronous communication. Moreover,

978-1-7281-4571-6/19/$31.00 ©2019 IEEE

because RSUs/routers naturally form into trees, caching within them efficiently disseminates data throughout the network without the need for a central arbiter. (iii) Data packets are secured regardless of their location. This means that data can be cached anywhere, and with no additional overhead can be relayed upon request. Furthermore, semantic naming helps automate key distribution in data authentication and access control, providing further avenues for usable security features.

III. Problem Statement and Related Works

In real world, due to the cost of deployment and support, RSUs can be deployed in many different environments with various densities and topologies [5].

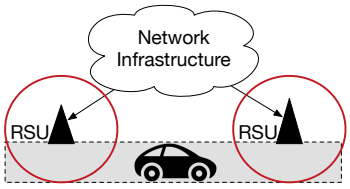


Figure 2: A Typical Topology

In this paper, we target the problem of degraded data downloading performance for vehicles in areas with limited RSU coverage. To illustrate our problem better, we use a simplified example scenario; that is, a vehicle drives out of a previous RSU's coverage, loses the RSU connectivity, and then drives into the next RSU's range (Figure 2). In addition, we separate the target applications into two categories. For non real-time applications, the problem is how to improve data downloading rate by fully utilizing the limited connectivity time. For real-time applications, the problem is how to ensure a consistent downloading rate for the user.

A. IP-based Prefetching

Content prefetching in mobile environments to reduce latency in data download has been well explored in a number of existing works [6]–[8]. In the context of vehicular networking, due to the intermittent connectivity between vehicles and RSUs, prefetching of data content becomes vital for ensuring the quality of information services.

Prior works such as [9]–[12] calculate the possible trajectory of the vehicle and content popularity, then prefetch different chunks of content to RSUs to maximize data retrieval rate. Fundamentally, RSUs fetch the required content from the cloud server based on a prediction model, and vehicles fetch desired content from RSUs when moving inside their signal range. The prediction and caching strategy aims to facilitate this content fetching process with minimum delay.

These approaches were proposed to run over IP, but its design lean towards a data-centric solution as their proposed mechanisms make data independent of the IP's point-to-point channels. Following this direction, it becomes ideal to apply a data-centric solution to the vehicular content prefetching problem to avoid the overhead of address allocation and connection state maintenance.

B. NDN-based Prefetching

The benefits of NDN for supporting producer mobility and securing content dissemination in vehicular networks have been well discussed in [1], [2]. Also, the authors of [13], [14] have discussed how a smart forwarding mechanism which makes use of NDN's data-centric asynchronous communication can benefit V2V information exchange in ad-hoc networks.

A number of content prefetching solutions using NDN have been proposed[1]. [15]–[17] focus on the optimal placement of content on RSUs based on travel path and content popularity, which relies on a predefined distribution model. [10] adopts a learning based approach for calculating the data popularity among vehicles, but also allocates data content onto RSUs based on historic requests.

NDN-based approaches replace IP-based communication with NDN-based asynchronous Interest-Data exchange to fit prefetching's content-centric nature. Also, most existing NDN-based prefetching solutions focus on the prediction mechanism (e.g., prediction by Linear Regression, Markov Chain, etc.).

C. Issues in the Existing Solutions

A common issue with existing IP-based and NDN-based prefetching approaches is that their performance is dependent on using a centralized server for accurate predictions, as inaccurate estimation causes waste in storage resources, and hurts download performance. In this paper, we propose a requester-initiated prefetching solution to better satisfy the real-time needs of vehicles, and increase the utilization of in-network storage. Our solution does not make predictions based on predefined models, neither does it depend on prior travel paths nor request history, but prefetches content on RSUs based on real-time user requirements in a timely manner.

An existing approach following a similar idea was [18], which allowed vehicles to signal the currently connected RSU when a handover process was going to happen, and notify the following RSU to prefetch the required content. Therefore, the protocol is not transparent to RSUs. Other than [18], existing approaches like [16] [10] also make use of RSU and infrastructure side routers for content caching. This means RSUs and wired infrastructure side routers need software upgrades to support new protocols, which increases deployment difficulties. Rather than relying on RSUs or infrastructure routers to cache content, in RapidVFetch, we take an alternative approach by making use of V2V communication to initiate content prefetch requests and content delivery. In doing so, we do not need to upgrade existing RSU infrastructure, and only require the vehicles involved in the communication process to support our protocol. This also reduces the attack surface on RSUs as we can authenticate the communicating vehicles through physical connectivity to the RSUs.

Another issue with existing work is the lack of security considerations in prefetching. The prefetching process is

[1]We also include works based on Information-Centric Networking (ICN), which represents a broad research direction. NDN is a realization of ICN.

asynchronous and how to ensure integrity, authenticity, and confidentiality of prefetched data becomes a main challenge. In RapidVFetch, we provide an approach to protect sensitive information in NDN names and Data content from eavesdropper (e.g., other vehicles, RSUs).

IV. An Overview of RapidVFetch

A. Assumptions

RapidVFetch does not make any assumptions on the topology being used in the V2X networking and does not require any centralized services to predict a vehicle's route or data consumption. In addition, RapidVFetch is transparent to RSUs and the infrastructure behind RSUs, thus minimizing the deployment complexity and requiring no overhead at RSUs and backbone routers.

RapidVFetch makes the following assumptions in its design.

1) **NDN Protocol.** RapidVFetch assumes that the vehicles, RSUs, and the service provider supports the NDN network stack and RSUs are capable of caching Data packets. However, we do not require the assumption that NDN has been fully deployed in the infrastructure; that is, the routers between RSUs and the service provider can be IP routers.
2) **Knowledge of RSUs.** We assume a vehicle has knowledge about the ranges of RSUs as well as the UUID of the subsequent RSUs on its traveling route. We argue the assumption is reasonable because RSUs are relatively static, that is, the change of RSU's range, UUID, and location is infrequent. This information can be collected, for example, by a map application.

B. A Design Overview

The goal of RapidVFetch is to facilitate and secure the data downloading from the infrastructure to vehicles by utilizing broadcast V2V communication and NDN's content-centric networking model. To be more specific, for non real-time applications, RapidVFetch aims to improve data downloading rate by fully utilizing limited connectivity time; for real-time applications, the goal is to keep the requester online and ensure a smooth downloading rate.

In a nutshell, as shown in Figure 3, a *requester* contacts a *forwarding vehicle* in the target RSU's range through V2V communication (which may contain single or multiple hops) to prefetch data (❶). For non real-time applications (e.g., video downloading), when the requester drives into that RSU's range, it can re-send Interests for the prefetched content and benefit from proactive caching (❷), improving utilization of the limited connectivity to RSU. In the case when real-time communication is required (e.g., online conferencing), the forwarding vehicle can also forward the replied content back to the requester in order to keep the requester online (❸). Importantly, RapidVFetch considers each Data's security in the downloading process. The downloaded Data packets are signed and can also be encrypted using NDN's built-in security primitives and name-based access control (NAC) [19].

By letting each requester initiate their own prefetch Interest packets, RapidVFetch always provides accurate prefetched data at target RSUs, getting rid of the centralized service for content/location prediction. In addition, this allows the requester to have a better control on when to trigger the prefetching, so RapidVFetch has more flexibility to cope with complicated road conditions (e.g., unexpected delay). For example, if there is a traffic congestion before entering the next RSU, the requester can delay the prefetch. At the same time, since a RSU does not need to treat prefetching Interests differently, the RSUs do not need to be aware of the RapidVFetch protocol. Consequently, RapidVFetch can be deployed at vehicles only without bothering RSUs and the infrastructure behind RSUs.

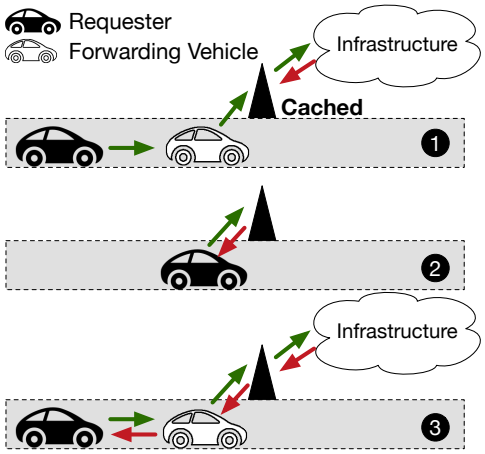


Figure 3: An overview of RapidVFetch

V. The Design of RapidVFetch

A. Basic RapidVFetch and Real-time RapidVFetch

In RapidVFetch, the requester contacts forwarding vehicles through V2V communication to ask them to forward Interest packets to target RSUs. Depending on the application use case and the capability of the forwarding vehicle, RapidVFetch supports two different versions to satisfy different needs.

- Basic RapidVFetch. Forwarding vehicles only need to forward lightweight Interest packets and do not need to listen to the relatively large Data packets. Therefore, the forwarding vehicle's overhead is minimal; for example, considering video streaming as a typical downloading use case, the video frame identifier is much smaller than the video frame itself. When the requester enters the RSU's range it will benefit from this precaching.
- Real-time RapidVFetch. Forwarding vehicles not only forward Interest packets but will also forward Data packets back to the requester. This works for real-time applications like multiplayer video games or online video/audio calls.

B. RapidVFetch V2V Protocol

RapidVFetch is initiated by the requester's Interest through a V2V communication channel. Specifically, when an RSU is not available, the requester will broadcast its Interest packets through V2V interfaces. The name of such an Interest

remains the same as the normal Interest packets sent from the application; however, the requester will perform several modification by RapidVFetch protocol. First, the Interest will carry a forwarding hint [2] which is the unique prefix of the desired RSU. As stated in the assumptions, such information is relatively static and can be learned in advance. Second, the Interest will be sent with NDNLP[3], which is a link layer protocol below the network layer (i.e., NDN) but above real link layer protocols (e.g., IEEE 802.11p). NDNLP is able to carry information for multiple purposes, e.g., fragmentation, failure detection, etc. In RapidVFetch, we introduce (i) a new flag in NDNLP to indicate whether such an Interest is for basic RapidVFetch fetch or real-time RapidVFetch fetch, (ii) a field to carry the requester's GPS information, and (iii) a field to carry the target RSU's GPS information.

These Interest packets will be sent to V2V broadcast media. One or more vehicles within range of the requester will hear the Interests. Once received, a vehicle V will parse the Interest packet along with its NDNLP header.

- V first compares the unique prefix of the RSU it is currently connected to with the RSU prefix obtained from the Interest's forwarding hint. If they match, it means V is under that RSU's coverage and is able to send out Interest packets to the desired RSU. In this case, in order to cope with potential conflicts, V will wait for a random time interval and then express these Interests to the RSU.
- Otherwise, V will first calculate whether it is closer to the RSU than the requester through the GPS information contained in NDNLP header and its own GPS coordinates to decide whether it is an eligible forwarder of the Interest. If it is, V will wait for a random time interval that is inversely proportional to the distance between the requester and V before it forwards the Interest [20]. If during this time interval a nearby vehicle transmits an Interest with the same name, the scheduled transmission of the Interest can be canceled to avoid duplicate transmission. After V forwards the Interest and overhears an Interest of the same name, V can treat this as an implicit acknowledgment that the forwarded Interest has been received and rebroadcast by one of its neighbor vehicles. Otherwise, V will retransmit the Interest based on its local policy.

As a result, the Interest packet will be forwarded by the forwarding vehicle that is farthest from the requester until it reaches the desired RSU.

In the case of basic RapidVFetch, each forwarding vehicle can simply erase the corresponding state in its local NDN forwarder so that it will not listen to replied Data packets. This reduces state overhead and saves bandwidth at forwarding vehicles. In contrast, forwarding vehicles in real-time RapidVFetch will each hold the pending Interest in its NDN forwarder, so that when the Data comes back it can forward the packet back to the requester.

[2]A forwarding hint identifies the gateway into the infrastructure but will not affect forwarding after that.

[3]https://redmine.named-data.net/projects/nfd/wiki/NDNLPv2

In basic RapidVFetch, when the requester moves into the target RSU, it can fetch all the prefetched content with minimum round trip time. Due to potential lack of forwarding vehicles and packet loss of wireless channel, some content might not be successfully prefetched (where the requester can notice it through round trip time measurement). Interest packets that do not hit the cache will be forwarded to the producer as per the regular V2I communication.

RapidVFetch does not make assumptions on the reliability of the underlying V2V and V2R media: when reliability is necessary, RapidVFetch can do Interest retransmission in case of timeout; as for real-time applications, Interests for contents that are within the accepted delay period will be retransmitted. Moreover, content will most likely be cached nearby when packet loss happens, so a retransmitted Interest does not need to go across the network again.

C. Securing Vehicular Data Prefetching and Downloading

In RapidVFetch, we consider the security of both the data prefetching process and the downloading process, which further includes (i) authentic use of RapidVFetch on RSUs, (ii) integrity and authenticity of downloaded data, and (iii) user privacy and the confidentiality of downloaded data.

1) Authentic Use of RapidVFetch on RSUs: It is nontrivial to authenticate users in an open wireless network system. To date, a number of vehicular security architectures have been proposed to employ a Public Key Infrastructure (PKI) in vehicular networks. However, these approaches provide means to ensure a vehicle has a valid plate, the driver has a valid license, or the vehicle is made by a verified manufacture, but they are not sufficient to prove the application is honest, e.g., whether the application will send spam or DDoS traffic to a RSU. Therefore, in this work, to mitigate spam Interest packets towards the RSU, we consider a simple but reliable means to authenticate a user: only forwarding Interest packets sent by users within RSU coverage.

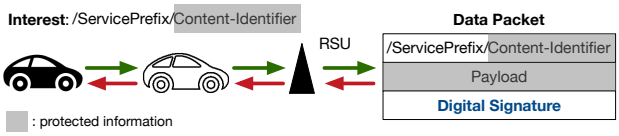
This feature also distinguishes our work from existing works. For example, [18] proposes a solution where a previous RSU issues an Interest packet to the next RSU. However, it is difficult for a RSU to verify that the Interest is from another RSU and is truly originated by a vehicle that will later drive into its own coverage. In contrast, RapidVFetch reduces the attack surface and RSUs can keep the strict access policy (i.e., traffic within coverage) without being aware of the RapidVFetch protocol.

2) Data Integrity and Authenticity: Acting as an Interest forwarder, the forwarding vehicle may direct the Interest to a fake service provider maliciously; as can the RSU, which may alter replied Data packets in their cache. Such alteration or fake Data packets will degrade the downloading and may cause damage to the requester. It is difficult for today's channel-based security model (e.g., TLS, QUIC) to ensure the data integrity/authenticity because the data will be cached at the RSU. This is because by the time the data is used by the requester, the secured channel does not exist any more and thus no security can be guaranteed.

However, empowered by NDN's built-in security, in RapidVFetch, every Data packet is signed (Figure 4) by its original producer, e.g., the video streaming service provider. Consequently, such alteration or fake Data packets will be detected by verifying Data packet signatures. If the data has a bad signature, the requester should resend the Interest packets and fall back to the normal downloading mode.

3) User Privacy and Data Confidentiality: A forwarding vehicle is able to know the Interest packets from the requester and the RSU can also grab information from the replied Data packets. This raises the user privacy concern for potentially leaking information of requester's ongoing actions and desired content. However, NDN names are not necessarily readable to eavesdroppers and the Data payload can also be protected through cryptographic techniques (Figure 4).

As a simple and effective solution to prevent information leakage, the requester and the service provider can negotiate a shared secret (e.g., through Diffie-Hellman protocol) asynchronously (e.g., ahead of vehicular communication). The shared secret can then be used to keyed hash or encrypt the sensitive name components in the Interest packets and content in the Data packets. For example, assuming the service provider's routable prefix is "`/service1`" and the full name prefix of the downloaded data is "`/service1/movie/movie1/frame`", the shared secret can then be used to hide the name components after "`/service1`" and the replied Data payload.



Each Data packet carries a digital signature. Sensitive information in Interest name, Data name, and Data content will also be protected.

Figure 4: Content-centric Security in RapidVFetch

A more systematic data confidentiality solution is to apply name-based access control (NAC) [19]. To be more specific, the service provider acts as an access manager who grants the application the access (i.e., decryption keys) to certain content. The access right granting is based on the service provider's policy and application's identity, which is represented by a public key certificate. By the NAC scheme, the content will be encrypted with encryption keys while the decryption keys will be distributed to authorized applications secretly (i.e., in ciphertext). In addition, different encryption keys protect data under different name prefixes, allowing a fine-grained and flexible access control.

D. Reliable Cache in Basic RapidVFetch

Since basic RapidVFetch utilizes target RSU's cache to provide better data downloading, in this section, we investigate the impact of RSU's cache on the RapidVFetch performance.

It is expected that a higher cache hit ratio will lead to a better downloading performance, because when the Interest cannot fetch the content from the cache, the Interest will be forwarded to the upper stream routers or the original producer, enlarging the round trip time. The hit ratio depends on several factors in practice, including the cache replacement policy, the time data is kept in the cache, the traffic condition, the freshness period[4] of data, etc. Given the complexity of real world settings, it is hard to model the hit ratio change into RapidVFetch. However, in this work we consider several factors that are independent from the environment – cache time and freshness period of desired Data packets.

To this end, first, the requester should try to shorten the cache time as much as possible. Assuming the round trip time t for all the prefetch data to be downloaded from the data producer to the RSU, It is optimal for the vehicle to trigger the prefetch at $T - t$, where T represents the time of entering the RSU. In this way, the data will be kept in the cache for zero time. Second, the data producer should properly configure the freshness period so that the prefetched Data packets can still answer the Interest packets when the requester enters the RSU's coverage. This can be achieved by several means; for example, the producer can set a longer default freshness period of Data packets it produces or the requester can estimate the time of entering the next RSU and notify the producer through the forwarding vehicle.

Besides these two factors, RapidVFetch performs better when RSUs enlarge their cache storage. Given the cost of hard disk storage is getting smaller as technology advances, we consider that the cache capacity will not be a limitation in real world deployment in the near future.

VI. Evaluation

In this section, we first compare RapidVFetch with existing prefetch approaches. We then simulate RapidVFetch against a baseline scenario where prefetch is not deployed and an optimal scenario where we assume the vehicle will always drive within an RSU's coverage.

A. Comparison With Related Works

We made a comparison between existing prefetching approaches with RapidVFetch in Table I.

As shown, unlike existing IP-based and NDN based approaches, RapidVFetch does not assume a known traffic pattern or content popularity distribution and is the first approach that does not require deployments at infrastructure (i.e., RSUs and the wired network). We did not compare our work with others through simulations because many of them utilize traffic/content prediction which highly relies on the environment settings and can be seriously affected by biased input, e.g., route prediction on a office worker who drives the same route everyday can be very different from that on a taxi driver whose route is nearly random.

B. Comparison With Baseline and Optimal Scenarios

Simulation Settings. To evaluate RapidVFetch's performance, we perform simulations over ndnSIM [21], a NS-3 based simulator. We simulated a 1200 meters single-direction road

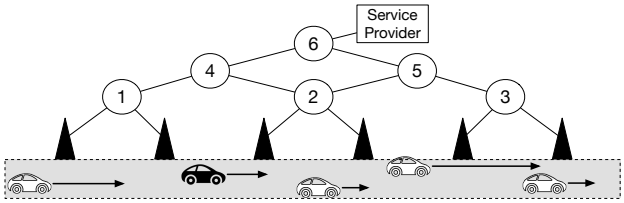
[4]Freshness Period is a TLV field supported by NDN. More details can be found in http://named-data.net/doc/NDN-packet-spec/current/data.html

Table I: A Comparison with Related Works

System	Main Approach	Cost	Transparency to the Infrastructure	Prefetch Accuracy
Perceive [15]	Prefetch by Centralized Service	Centralized vehicle info gathering and mobility/content prediction	✗	≈100% when mobility prediction is correct.
ADePt [10]	Learning Based Prefetch	Information gathering and decision making on each RSU	✗	≈80% when content popularity is known
Entropy-based Proactive Caching [16]	Markov Based Prediction	Centralized Training & Prediction, Cache Redundancy	✗	≈100% when router tree topology is 8 layers high
Optimal Content Prefetching [17]	Integer Linear Programming Based	Each RSU prefetches data based on retrieval probability assessment	✗	≈85-90% when content popularity and user mobility are known
PCNDN [18]	RSU Driven Prefetch	Negotiation Between RSUs. Allowing remote control of RSUs.	✗	≈100%
Our work	Requester Driven Prefetch	Distributed V2V Communication	✓	≈100%

with 6 RSUs placed at 100, 300, 500, 700, 900, 1100 meter respectively. Each signal-free zone between two adjacent RSUs is 80 meters. The requester starts at 0 meter with a velocity at 20 m/s. To be more specific, we simulated RSUs over IEEE 802.11b 2.4GHz with a signal range of 60 meters and V2V over IEEE 802.11a 2.4GHz with a signal range of 40 meters. In our simulations, the packet loss in wireless media depends on the distance between the transmitter and the receiver. The topology is a simplified tree as shown in Figure 5. We have disabled all the NDN cache of non-RSU nodes so that a requester cannot benefit from the cache in common ancestor routers.

In all the simulations, the requester will fetch Data packets one after another, so the performance is not related to Interest sending policies (e.g., batched Interests or parallel Interests). An Interest packet will be retransmitted in the case of timeout.

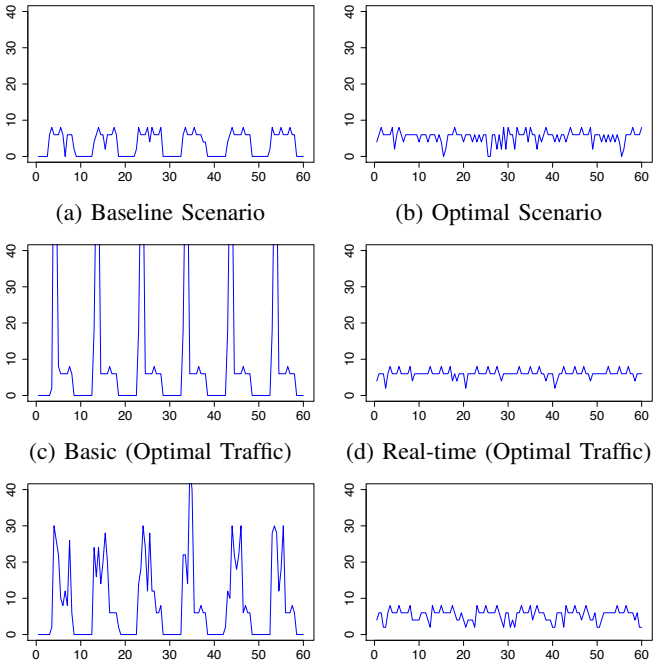


The cache of router 1-6 has been disabled.

Figure 5: The Topology For RapidVFetch Evaluation

Baseline and Optimal Scenarios. We first evaluate a baseline scenario (Figure 6a) where vehicles download data without RapidVFetch and an optimal scenario (Figure 6b) where we temporarily enlarged the RSU coverage to 120 meters so that the requester is always under RSU coverage. As shown, in baseline scenario, the downloading speed is zero when the requester is out of RSU coverage and back to full speed (about 8 pkt/s) after the requester enters RSUs. In the optimal scenario, the downloading speed is always around full speed. Fluctuations in downloading speed are caused by the packet losses in the wireless media.

RapidVFetch with Optimal Traffic. We then simulated both basic and real-time RapidVFetch (Figure 6c and Figure 6d) under an optimal traffic scenario where we temporarily added forwarding vehicles that are always available to the requester.



(a) Baseline Scenario (b) Optimal Scenario

(c) Basic (Optimal Traffic) (d) Real-time (Optimal Traffic)

(e) Basic (Suboptimal Traffic) (f) Real-time (Suboptimal Traffic)

Figure 6: Baseline, Optimal, and RapidVFetch Performance (x axis: simulation time (s); y axis: Downloading Speed (pkt/s))

Compared with the baseline scenario, the downloading speed in basic RapidVFetch is greatly improved: after the requester enters a RSU, there is an obvious increase in the downloading speed; this is because the requester can fetch pre-cached packets from the RSU with a minimum delay (fetch within one hop). After that, the downloading speed falls down to a normal downloading speed; this is because all the prefetched packets have been retrieved by the requester and later downloading subjects to the normal RTT. Regarding the real-time RapidVFetch in Figure 6d, the requester can keep consistently downloading content while moving out of the RSU coverage, so the download speed is close to that of the optimal scenario.

RapidVFetch with Suboptimal Traffic. We also simulated basic and real-time RapidVFetch (Figure 6e and Figure 6f) under a suboptimal traffic scenario when there may not be

available forwarding vehicles from time to time. To be more specific, we simulated 9 other vehicles on the road for forwarding packets, each with a random velocity and a random starting position following a Gaussian Distribution model with mean value of 20 m/s (standard deviation is 5 m/s) and 0 meter (standard deviation is 20 meters), respectively. As shown, compared with Figure 6c (upper downloading speed exceeds 40 packets/s) and Figure 6d, the downloading speeds in Figure 6e and Figure 6f become lower because when forwarding vehicles are not always available, fewer packets are cached in basic RapidVFetch and fewer packets are forwarded back to the requester in real-time RapidVFetch. Compared with Figure 6c, in Figure 6e, since a number of packets were not cached scattered among cached packets, the fluctuations in downloading speed are more spread out and the average downloading speed is lower.

VII. Discussion

A. A Large Gain Low Cost Application Model

RapidVFetch takes advantage of V2V communication in data prefetching in order to provide large gains in performance with few associated costs. RapidVFetch is transparent to RSUs, and as such both benefits as much as possible from incremental deployment and does not require any infrastructure changes – minimizing overall costs. Additionally, the cost for any vehicle to help another is minimal: it only requires forwarding interest packets, which are generally quite small in terms of overhead. Finally, if a future deployment could benefit by encouraging users to participate in prefetching, an application built over RapidVFetch could record each vehicle's contribution and incentivize prefetching (either monetarily or with higher priority in the app). Regardless of implementation details, RapidVFetch provides a platform that applications can easily use to speed up downloads and reduce down-time without changing the infrastructure.

B. Performance of RapidVFetch in Real World

RapidVFetch's performance relies on the road condition (i.e., number of forwarding vehicles nearby). If one considers three typical environments: urban, sub-urban, and rural, RapidVFetch will have the best performance (i.e., close to the optimal scenario) in urban districts, a lower performance in sub-urban areas, and performs worst (i.e., degraded to baseline solution) in rural areas. However, RapidVFetch is promising in practice because the deployment of RapidVFetch does not require any change to RSUs and the infrastructure behind.

VIII. Conclusion

RapidVFetch fully utilizes the high throughput and short connectivity time with RSUs by prefetching future data into target RSUs. For real time applications, RapidVFetch also allows a forwarding vehicle to be a data forwarder in order to achieve more stable data downloading. Through our work, we show that NDN as a data-centric network architecture fits better into vehicular networking for its support of asynchronous communication, in-network cache, and built-in security. Additionally, we show that, compared with existing works, RapidVFetch is able to achieve a better performance and get rid of the dependency on centralized services.

References

[1] G. Grassi, D. Pesavento *et al.*, "Vanet via named data networking," in *2014 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*. IEEE, 2014, pp. 410–415.

[2] L. Wang, R. Wakikawa *et al.*, "Data naming in vehicle-to-vehicle communications," in *2012 Proceedings IEEE INFOCOM Workshops*. IEEE, 2012, pp. 328–333.

[3] L. Zhang, A. Afanasyev *et al.*, "Named Data Networking," *ACM SIGCOMM Computer Communication Review*, 2014.

[4] Z. Zhang, Y. Yu *et al.*, "An overview of security support in named data networking," *IEEE Communications Magazine*, vol. 56, no. 11, pp. 62–68, 2018.

[5] A. B. Reis, S. Sargento, and O. K. Tonguz, "On the performance of sparse vehicular networks with road side units," in *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*. IEEE, 2011, pp. 1–5.

[6] Z. Jiang and L. Kleinrock, "Web prefetching in a mobile environment," *IEEE Personal Communications*, vol. 5, no. 5, pp. 25–34, 1998.

[7] G. Cho, "Using predictive prefetching to improve location awareness of mobile information service," in *International Conference on Computational Science*. Springer, 2002, pp. 1128–1136.

[8] A. J. Nicholson and B. D. Noble, "Breadcrumbs: forecasting mobile connectivity," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*. ACM, 2008, pp. 46–57.

[9] P. Deshpande, A. Kashyap *et al.*, "Predictive methods for improved vehicular wifi access," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, 2009, pp. 263–276.

[10] D. Grewe, S. Schildt *et al.*, "Adept: Adaptive distributed content prefetching for information-centric connected vehicles," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*. IEEE, 2018, pp. 1–5.

[11] Z. Hu, Z. Zheng *et al.*, "Game theoretic approaches for wireless proactive caching," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 37–43, 2016.

[12] R. Kim, H. Lim *et al.*, "Prefetching-based data dissemination in vehicular cloud systems," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 1, pp. 292–306, 2015.

[13] M. Meisel, V. Pappas *et al.*, "Listen first, broadcast later: Topology-agnostic forwarding under high dynamics," in *Annual conference of international technology alliance in network and information science*, 2010, p. 8.

[14] G. Grassi, D. Pesavento *et al.*, "Navigo: Interest forwarding by geolocations in vehicular named data networking," in *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2015, pp. 1–10.

[15] D. Grewe, M. Wagner *et al.*, "Perceive: Proactive caching in icn-based vanets," in *2016 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2016, pp. 1–8.

[16] N. Abani, T. Braun *et al.*, "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] G. Mauri, M. Gerla *et al.*, "Optimal content prefetching in ndn vehicle-to-infrastructure scenario," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 3, pp. 2513–2525, 2016.

[18] Y. Rao, H. Zhou *et al.*, "Proactive caching for enhancing user-side mobility support in named data networking," in *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, 2013, pp. 37–42.

[19] Z. Zhang, Y. Yu *et al.*, "Nac: Automating access control via named data," in *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 2018, pp. 626–633.

[20] L. Wang, A. Afanasyev *et al.*, "Rapid traffic information dissemination using named data," in *Proceedings of the 1st ACM workshop on emerging name-oriented mobile networking design-architecture, algorithms, and applications*. ACM, 2012, pp. 7–12.

[21] S. Mastorakis, A. Afanasyev, and L. Zhang, "On the evolution of ndnSIM: an open-source simulator for NDN experimentation," *ACM Computer Communication Review*, Jul. 2017.