Wireless Personal Communications (2020) 113:139–166
https://doi.org/10.1007/s11277-020-07182-9

CCVNet: A Modified Content-Centric Approach to Enable Multiple Types of Applications in Vehicular Networks

Rojin Tizvar[1] · **Maghsoud Abbaspour**[1]

Published online: 24 January 2020

Abstract
Intelligent transportation systems and connected vehicles, by utilization of new facilities and technologies, have a significant role in improving the quality of today's transportation. A wide range of applications can be defined in a vehicular network that each one has its own requirements. Compared to other wireless networks, vehicular networks face particular challenges due to the continuous and high-speed movement of vehicles. Concerning the characteristics of content-centric networking (CCN), it has the potential to deal with many of these challenges. In this paper, we modify the basic model of the CCN with the aim of supporting many types of application including safety and non-safety. The applications with pull and push behaviors are both considered in the design process. A new concept of shared-content among different applications is introduced and also taken into account in the presented model. To achieve these goals, the basic content retrieval mechanism has been expanded. The simulation results indicate that the new model has good performance in terms of throughput and content retrieval time.

Keywords Vehicular network · Content-centric networking · Content retrieval · Content naming

1 Introduction

In recent decades accidents, urban traffic and air pollution caused by fuel consumption are some of the biggest issues facing both developed and developing countries all around the world. So, improving the current transportation systems can have a significant role in improving the quality of people's lives [1, 2]. Utilization of new facilities and technologies can be an effective way to achieve this goal. Today vehicles that come with a variety of sensors can collect data from the environment and adding wireless communication capability, allows them to be connected and interact with other vehicles or the environment. Based

✉ Maghsoud Abbaspour
maghsoud@sbu.ac.ir

Rojin Tizvar
r_tizvar@sbu.ac.ir

[1] Faculty of Computer Science and Engineering, Shahid Beheshti University, G. C. Evin, Tehran, Iran

on the categories provided for the Internet of Things (IoT), transportation will be one of the main applications in the future [3–5]. In addition, considering the definitions provided for UrbanIoT, intelligent transportation systems and connected vehicles, can also be used as the main resources in a smart city [5, 6].

An intelligent transportation system enables a wide range of application domains. These applications can be grouped into three types, each with its own performance requirements: the main objective of safety applications is to reduce the risk of accidents and injuries; traffic management applications try to facilitate the traffic management process by controlling and improving the traffic flow; as its name implies, Infotainment (information and entertainment) application provides users with the internet access and additional informative or entertainment services like file sharing or video streaming [1, 7–10].

Despite the variety of suggested communication technologies like DSRC/ WAVE, IEEE 802.11p protocol, cellular network standards (LTE, LTE-A, and 5G), WiFi, and WiMax [1, 7, 10–14], until now, there is no comprehensive and perfect solution that can provide the requirements of all types of vehicular communication. Compared to other wireless networks, vehicular networks face special challenges due to the continuous and high-speed movement of vehicles. Rapidly changing topology and harsh wireless propagation environment, which result in short-lived and intermittent connectivity, are unique features of vehicular networks. From the networking perspective, high reliability of the system being required in packet delivery, as well as the ability to transmission with the least possible delay are major challenges in designing a vehicular network with the explained features [1, 15, 16]. Lack of a centralized control unit, intermittent connections, and dynamic topology are all the traits of vehicular networks that can challenge the use of TCP/IP architecture to make a stable connection between vehicles [10, 15, 17, 18]. By decreasing the need to establish and maintain an end-to-end connection, content-centric networking (CCN) can give better performance than TCP/IP in this type of networks [15, 17, 19]. Unlike the IP architecture, there is no prior need for network parameters configuration like IP addresses and subnet masks in CCN [16]. Furthermore, since the CCN model intrinsically supports content consumer mobility, it is well suited for such dynamic networks. According to the content delivery process of CCN, as it is compared in [19], with the increase in vehicles speed CCN performs much better than mobile IP. The CCN can also well benefit from the broadcast wireless channels nature. By multipath forwarding of interest and data packets, helps and speeds up the content sharing between vehicles. On the other hand, it is possible to overcome the problems caused by vehicle mobility and intermittent connections by distributed in-network caching feature [10, 15, 16]. Concerning the characteristics of CCN, it has the potential to deal with many vehicular networks challenges described above. In addition to all these motivations, all types of application in vehicular networks are intrinsically content-centric. The main goal of any application defined in this network is to get access to particular content or information; for instance, the traffic information of a street or announcing a car accident. In these situations, the identity of the content provider is not important for the application or the user.

Beside the CCN advantages, utilizing it for a vehicular network has its challenges and limitations. Many researchers have applied CCN architecture in vehicular networks, and they approved its suitability for this kind of network. Most of the works have made different modifications in the content retrieval mechanism, to improve the performance of basic CCN in a vehicular network. Despite overcoming many limitations, there are some challenges that still exist. Although the building blocks of CCN are basically defined, the content naming scheme, the content retrieval mechanism, and caching policies should be specifically designed for the network and the application

requirements. A well-designed architecture should be able to support different types of application that can be defined in vehicular networks. Furthermore, because of the native receiver-driven model of CCN, a content-centric vehicular network faces the challenge of how to support push model data transmission. This is crucial for safety applications. The network has to provide a suitable interface for application developers, so they can efficiently develop new applications on top of the CCN model.

In this work, we extend the basic model of CCN for vehicular network communications to cope with the existing challenges. Some modifications in the CCN building blocks are made to target the following objectives:

- Supporting many types of application including safety, traffic management and infotainment;
- The content retrieval mechanism is extended in order to support push model communication;
- To improve the content retrieval, a new concept of shared content among different applications is introduced and considered in the design process. By defining two different data structures any application developed on top of this architecture, can use both shared and application-specific contents;
- The presented naming scheme imposes no hard restrictions for content naming on the application developers and also provides them the ability to the geographical and temporal targeting of a content.
- To make the network capable of distinguishing between different packet types including Interest, Data, Push, Pull, Shared or Application-specific, a new packet header is designed;
- To overcome the challenges of cache sharing by multiple applications, the new architecture includes two different content store types, one for each application and a central one for shared contents.

The remainder of the paper is organized as follows. In Sect. 2 the content-centric network concept is described in a nutshell and after that, some related works in this field are briefly reviewed and the motivations of this article will be indicated. Our work is completely explained in Sect. 3 and in Sect. 4 the performance of the presented work is evaluated. Finally, we conclude the paper in Sect. 5.

2 Background and Motivation

The model of CCN architecture is highly suited for time and location-relevant applications; hence, it naturally matches vehicular network applications. Named content retrieval, name-based routing and forwarding and in-network caching are the key features of content-centric networking that make it an appropriate solution to meet the challenging demands of a vehicular network communications. In this section after briefly describing the content-centric networking and its main features, some researches in the field of content-centric vehicular networks are reviewed.

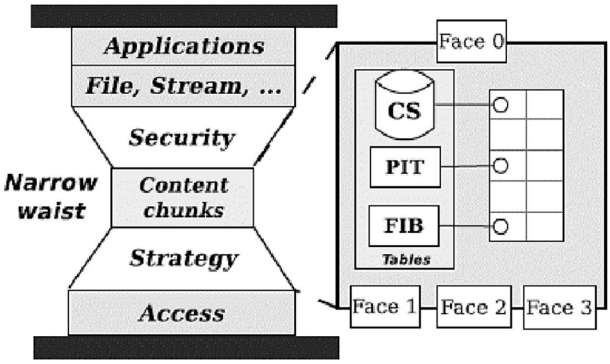

Fig. 1 The CCN node basic model [24]

2.1 CCN Architecture: An Overview

Nowadays, people value the internet for the content they can access through it, but with the current architecture, the communications are based on the content's location. Content-centric network (CCN) paradigm emerged to replace the content itself with its location. In other words, it makes a shift from the host-centric conversational model to a content-centric model [16, 20].On these networks, every chunk of content, called Named Data Unit (NDU), has a unique and persistent name and can be individually stored and accessed through the network. So, instead of making a connection to a host or server that owns a content, a user sends his/her request to the network; consequently, the network as the main service provider retrieves and delivers the requested content to the user. It is worth noting that due to the in-network caching feature in the new model; the desired content can be retrieved from any intermediate nodes in the network and not only the original producer [16, 20–23]. To sum up, understanding the content the user has requested for, finding a node that can provide the content, and the routing and forwarding mechanisms are all network responsibilities.

With the aim of changing the current internet architecture, Van Jacobson and his team at PARC research center presented the CCN architecture [20].This architecture was later known as the NDN project at UCLA University. As can be seen in Fig. 1 the CCN inherited the hourglass model of IP architecture, but the IP address is replaced with the content chunks. Some of the main advantages of CCN architecture over IP for IoT applications are presented in [3, 17, 21, 23]. Same as other *information-centric networks (ICN)* [21, 22], CCN has four main building blocks: *content-based naming, content discovery and delivery, in-network caching and content-based security*. The primary concept and characteristics of each block have been defined in [20]. According to the model, as shown in Fig. 1, each network element has three main data structures: Content Store (CS), Forwarding Information Base (FIB) and Pending Interest Table (PIT).

2.2 Related Work

After understanding the model of content-centric networks and explaining its potential benefits in vehicular networks, in this section, some related works in this area are briefly

reviewed. It should be noted that this article has no focus on the Security of CCN, therefore, the related work in this area are not mentioned.

In a content-centric vehicular network, a vehicle broadcasts an *interest*[1] packet, which includes the name of the desired NDU. Each neighboring vehicle received the packet checks its CS. If a copy of the NDU exists, as a content provider, it will create and broadcast a *reply (data)*[2] packet that contains the requested content. If not, it sets its PIT and forwards the packet. The interest travels through the network until it reaches a content provider or producer. Then, the reply packet will be forwarded toward the original requester (consumer) by any vehicle that has a corresponding entry in its PIT. During this process, any intermediate vehicle may cache the NDU in their CS. In this model the content producer is the original source that generated the content and the provider would be any node or vehicle that has a copy of the content in its CS.

Content name is composed of one or more variable length strings, separated by '/', and the naming scheme has a hierarchical structure like URIs or IP addresses. Some basic conventions for the name structure have been defined in CCN, but the name's semantics and the number of substrings are application-specific [15, 20, 23]. Wang et al. [25] have focused on the data naming for a traffic information application. Based on the provided naming design requirements, they have proposed a naming, which is *'/traffic/geolocation/timestamp/datatype/nonce/'*. In [26], a prototype called V-NDN has been implemented and tested on the UCLA vehicular testbed. They have also considered traffic application (road photo and traffic info) and used a similar hierarchical naming scheme for naming the traffic information of a particular location. Yu-Ting Yu implemented NDN as an overlay on top of IP network and used IP addresses for vehicles identification and neighbor discovery [17]. They used *'application_id/data_ object_id/chunk_id'* naming structure to identify the application and the desired content chunk. In [27] the authors have added "/high" and "/low" substrings to the names to apply priority between packets.

Routing protocols in content-centric networks can be either proactive or reactive. In proactive ones, control messages are periodically sent by content producers to keep the FIB tables up to date. Since contents are generated related to time or location and in a distributed manner, reactive strategies and flooding are usually used in a content-centric vehicular network. In this case, the content retrieval process begins with interest flooding [10, 15].

Interest flooding is the simplest way to transfer content requests in wireless channels and a suitable solution for dynamic networks as well. On the other hand, considering the broadcasting, vehicles can overhear the content (Data packets) requested by others and if needed store it in their CS. However, flooding will lead to the broadcast storm problem [28]. Therefore, it is important to control broadcasts in order to avoid this problem as much as possible [7, 10, 15, 16, 29]. A set of timers is used in the work presented in [30] in which the authors have considered traffic management applications in a vehicular network. A defer timer is used to improve broadcasting by reducing the collision probability. The same approach is used in CCVN [31] and [27]; two timers are used to apply a priority between packets. Arnould and cooperators considered vehicles equipped with multiple network interfaces, including 802.11p, WiMax and cellular [32]. In this article, the interest is sent over the interface with the least delay; in contrast, in [26] packets are sent over all existing interfaces. In [26, 30, 33, 34] a timer is defined by which the node that has the maximum distance from the content

[1] In this paper we use request and interest interchangeably.

[2] In this paper we use Data and Reply packet interchangeably.

consumer will forward interest or data packets before other neighboring nodes. By defining a new parameter named *Interest Satisfaction Rate (ISR)*, another forwarder selection mechanism is presented in [29]. Some kinds of content provider selection mechanisms are proposed in [31, 34, 35]. They all have considered non-safety applications in vehicular networks. According to the native receiver-driven model of CCN there is no primary supporting of push messages in content-centric networks, in [36] a proactive data dissemination scheme for pushing critical content to one-hop neighbors is presented.

As stated before, vehicular networks can benefit from the in-network caching feature of the content-centric model. Defining the appropriate content storing and replacing policies is the main issue for using this feature. Based on a predefined policy, when a vehicle receives a content chunk from the network, should decide about caching or dropping the chunk. In [33] three different caching policies are compared: (1) no caching mechanism is used, (2) the vehicles cache all the chunks they receive from the network (aggressive caching policy) and (3) peer-only caching in which only the peers sharing the same application may cache the content. Grassi et al. [26] have also considered aggressive caching policy in their work.

Previously mentioned, the vehicular network applications can be grouped into three types: safety, traffic management, and infotainment. According to application requirements, they can also be grouped as *safety* or *non-safety* services. In [7] list of the requirements for the use cases of both services are provided. Most of the works presented above have considered a non-safety and often traffic-management application in a content-centric vehicular network; this stemmed from the nature of this type, which completely matches the CCN receiver-driven model. For example, the common task of a file sharing application or a car navigation application is to send a request for specific information or content and get the desired content in return. It means the content is *pulled* by the content consumer. By contrast, there are many applications of an intelligent transportation system that fall into the safety group and often have event-triggered or periodic behavior. Consider a car accident warning application (or any event-triggered application) and an application in which the speed of the vehicle is sent periodically to the network, in both no prior 'interest' will be sent. In other words, in these use cases, the content is *pushed* by the content provider to the network. In addition, in none of the related work described in the previous section, all types of vehicular application are considered together.

According to the research challenges provided by IRTF [37], the CCN network should provide a suitable interface for application developers, so they can efficiently develop new applications on top of the CCN model. Another issue mentioned in [37] is that besides the need to support push model in content-centric networks for IoT applications, by sharing a single CS across different applications there may be technical and business challenges. For instance, using a single limited-size CS for all applications with different memory requirements, will cause starvation; this will be crucial for safety applications.

In a vehicular network, there are many types of content that can be shared among different applications. The contents like weather condition, traffic information, or occurrence of a car crash can be measured, generated and provided by many of the network elements; on the other hand, these contents can be also used by many applications. So, by considering this kind of contents in the network design, the performance of CCN content retrieval mechanism can be improved.

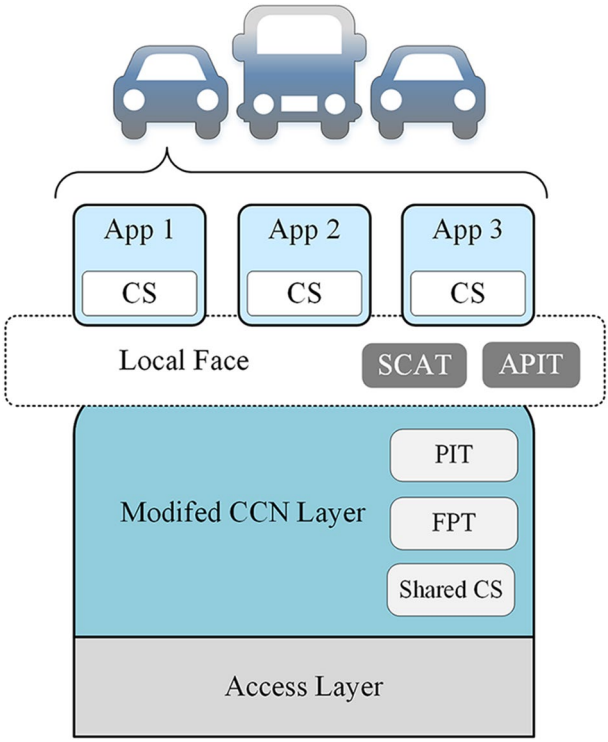

Fig. 2 CCVNet architecture for vehicular networks

3 Proposed Work

In CCVNet we modified the basic model of CCN to support many types of application in a vehicular network together. The applications with both *pull* and *push* behaviors are considered in the design process. A new concept of shared content among different applications is introduced and also taken into account in the new model. The naming scheme and the modifications in content retrieval and caching mechanisms are respectively presented in Sects. 3.2, 3.3 and 3.4. The overall changes made in the CCN model for the OBU (On Board Unit) used in a vehicular network is represented in Fig. 2. It should be noted that in this work, vehicles are assumed to have one network interface. For this reason the FIB table of the CCN can be omitted because according to the wireless communication and broadcast transmission, the forwarding block of each node will only decide whether to broadcast the packet over the single network interface or not.

3.1 Shared-Content Concept

Like the TCP/IP architecture, many applications may be installed on top of the CCN network layer. Many types of content can be shared among various applications. Consider two different applications like google map and Waze. When the user wants to use them, at first, they should load their own map tiles in the application. When the map is loaded, they want to provide some information like road traffic for the user. This traffic information is

Table 1 An example of shared content for applications table (SCAT)

Shared content id	Application id
SC type 1	App1
	App2
SC type 2	App3
	App2
SC type 3	App2
	App4

identical in both applications. From the perspective of a content-centric vehicular network, this content is shared between these two applications; this can be generalized to many other contents like weather condition, vehicles speed, empty parking spaces or any other data generated by vehicular sensors. Therefore, we will have both *application-specific* content and *shared* content. Shared content can also be generated by various applications. For example, each vehicle that has the temperature sensor can generate the weather condition information. In contrast, a content like a post in any social network application can only be generated by the corresponding application.

By considering shared contents in the network, various applications can be supported in the network and there is no need to install all of them on each vehicle. In addition, they are not forced to use identical naming scheme for their contents in the application design. An application can also request and consume a data generated by another application without knowing its naming scheme; it should just request for this shared content and obey the network provided standard for the data naming. This will be more explained in the following section.

In a content-centric vehicular network, there will be many types of shared content, and each should be identified by a unique identifier (*id*). Any application installed on top of this network, besides its own app-specific content, should determine all types of shared content needed for its functionality. Therefore, as illustrated in Fig. 2 there is a local interface between the applications and the network layer that support application registration. An application registers with the types of required shared contents. This information is stored in a table named **SCAT** (*Shared Content for Applications Table*). Table 1 shows an example of SCAT for three types of shared content. Shared content can also be pulled from or pushed to the network. For instance, an application may send a request for a traffic content or may send a packet that announces the arrival of an emergency vehicle.

3.2 Content Naming

The name is the most important attribute of an NDU (content chunk) because the content retrieval mechanism is completely based on it [18]. Some naming design requirements are provided by [25] including *Geographical Scoping, Temporal Scoping and Duplication Detection*. They have declared that these requirements are derived from the traffic information dissemination application and any naming structure should be able to accommodate different data naming structures for individual applications. Hence, our design imposes no hard restrictions for content naming on the applications. Considering that there are two types of content, the naming design for each is as follows:

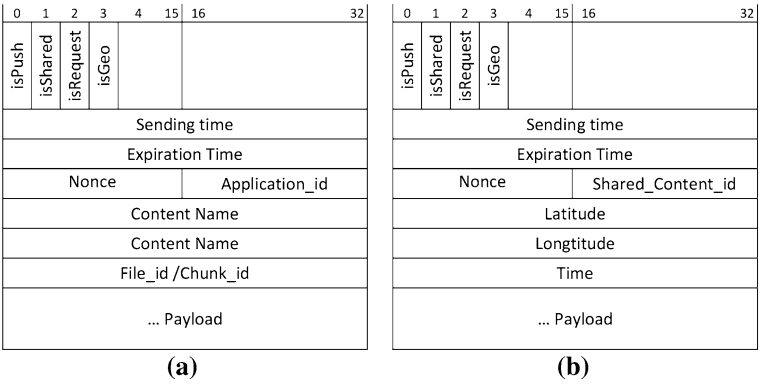

Fig. 3 The interest and reply packets format in CCVNet. **a** Application-specific content, **b** shared content

- *Application-specific contents* Because the request and reply are both sent by the same application, the naming details and substrings are defined and can be understood by the application itself. The network should just know the application identifier (App_id) to be able to discover and deliver the requested content. For instance, the content naming for a music application can be something like *App_id/singername/album/track/chunk1/* or for another application, it can be *App_id/folder1/folder2/file_type/file_id/ chunk_id/*. To sum up, from the network point of view the application-specific content name is *App_id/Custom/*. According to the fact that in a vehicular network many applications may need a geographical targeting, a flag named isGeo is added to the packet header, which may help the applications and will be more explained in Sect. 3.3.
- *Shared contents* Name of shared contents should be standardized by the network because they are shared among applications and need to be understood by all of them. Usually, geolocation and timestamp are both important attributes of shared content, for this reason, we suggest *SharedContentTpye/Geolocation/Timestamp* for the naming of this type. For example, the traffic information of a specific location can be named by *traffic/latitude/ longitude/time* or a car accident occurrence can be named by *accident/latitude/ longitude/time*. It is worth noting that any new shared content type and its naming standard can be added to the network but in the applications registration process it should be clarified for any interested developer. A public announcement must also be made so that anyone can develop an application that works with the new shared content type.

3.3 Content Retrieval Mechanism

In a content-centric vehicular network, the main responsibility of the network layer is to deliver the content from the content provider to the content consumer. When the consumer needs a content chunk, it sends a request to pull the desired content from the network. In addition, in some cases like accident occurrence, the content provider needs to push content to the network. To support the delivery of pushed and pulled contents and manage the shared content between different applications, some modifications in the CCN content retrieval mechanism are needed. First of all the information required for content retrieval process should be formatted. The proposed *packet header* is 25 bytes and consists of 11 fields and is illustrated in Fig. 3 The fields of this header can be described as follows:

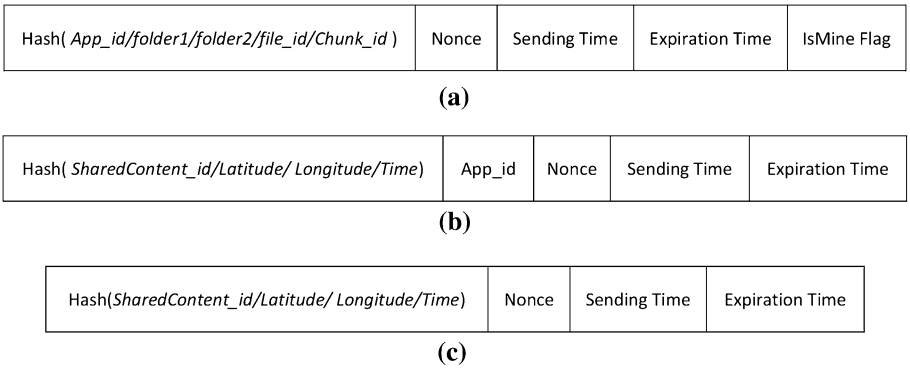

Fig. 4 The structure of an entry in **a** PIT table, **b** APIT table and **c** FPT table

- *isPush* When this flag is set, that means the packet is pushed to the network.
- *isShared* This flag must be set in any kinds of shared content packets.
- *isRequest* This flag is 1 for interest packets and 0 for reply (data) packets.
- *isGeo* when this flag is set by an application, the packet header contains the target geolocation information (latitude and longitude), starting from the 14th byte in two blocks of 4bytes. This information can be used by the network for better routing and forwarding.
- *Sending time* The time on which the packet is sent will be added in this field.
- *Expiration time* This field determines the duration of the interest lifetime.
- *Nonce* This field is a random number that makes it possible to detect duplicates. In addition, in this work, the nonce in the reply packet of a satisfied interest will be exactly set to the nonce in the corresponding interest packet by the content provider.
- *Naming* 14 bytes are provided in the packet header, to embed the content names given by the applications. As it can be seen in Fig. 3a, b these fields are set to different values in application-specific and shared packets as described in the previous section. As stated above shared contents' name are standardized by the network. To make the packet format identical for both content types, we have considered a confined space for the naming field, but for application-specific names a TLV (Type-Length-Value) format can also be used.

In addition to the **PIT** table in the basic model, another table named **APIT** (Applications PIT) has been added to the CCVNet. Two applications on a vehicle may individually send two similar interests for an identical shared content chunk. In order to send just one interest to the network for both of them, the APIT is added to the local face (Fig. 2). For each interest, a new entry will be inserted to the APIT which contains the App-id of the applications. So, in CCVNet for many entries in the APIT that are interests for the same content chunk, only one interest will be sent to the network and stored in the PIT. When the vehicle receives the data packet, the local face checks the APIT and send the data to the all related applications. Another table called **FPT** (Forwarded Push Packets Table) is added to the model. By storing the already forwarded push packets in this table, we avoid retransmission of the same packets and reduce the redundancy and the effects of broadcast storm. Figure 4 represents the entry structure of these three tables. It should be noted that the expiration time field specifies the interest lifetime

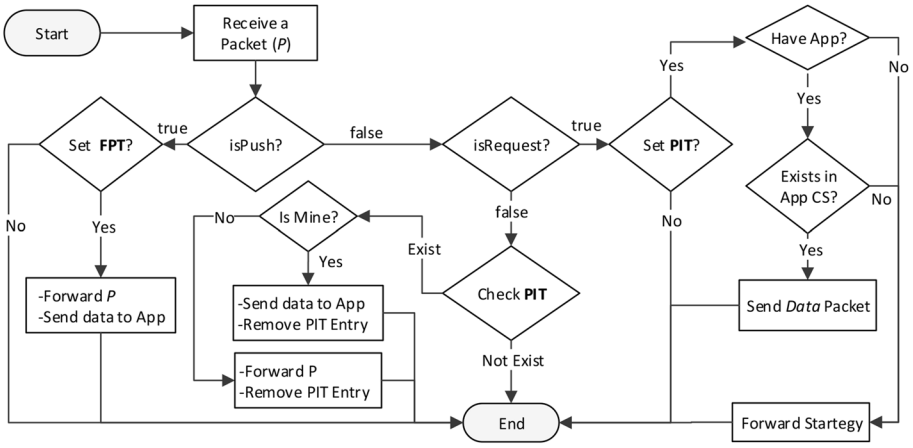

Fig. 5 The flowchart of the content retrieval mechanism for application-specific contents. The "set" function of PIT and FPT tables will add the new entry to the table and return Yes if the entry has been already added to the table will return No

and the entries that have been expired would be periodically removed from the PIT, the APIT, and the FPT.

When a shared content is pushed to the network and received by a vehicle, its type is determined from the packet header. Afterward, the local face checks the SCAT table and finds the applications that are interested in this type and sends the content to each of them. According to the content types and application behaviors (push or pull), four situations may happen. Content retrieval mechanism for each one is as follows:

- *Application-specific content (pull)* The application creates an interest packet. The interest will be added to the PIT and send over the network interface. This packet is forwarded by the neighboring nodes, which do not have the requested content. The complete retrieval mechanism flowchart is illustrated in Fig. 5. The forwarding mechanism is based on flooding, but to reduce the broadcast storm effect a probabilistic approach is used. So, a random variable named *Forward Probability* is defined, which reduces the number of neighboring nodes that forward the packet simultaneously.
- *Application-specific content (push)* The push packet is created by the application and after setting the FPT is sent over the network interface. Figure 5 shows each node reaction to the reception of this packet.
- *Shared content (pull)* The application creates an interest packet. The interest will be added to the APIT and if needed to the PIT and send over the network interface. The complete retrieval mechanism is illustrated in Fig. 6. Any application that works with this shared content type may be able to generate the requested content and the data packet.
- *Shared content (push)* The process is the same as the application specific contents, except that on reception of this packet the SCAT will be checked as presented in Fig. 6.

In addition to these flowcharts (Figs. 5, 6), the pseudocodes of the retrieval mechanisms for Application-Specific and Shared contents are also presented in the Appendix section, in Algorithm 1 and Algorithm 2 respectively.

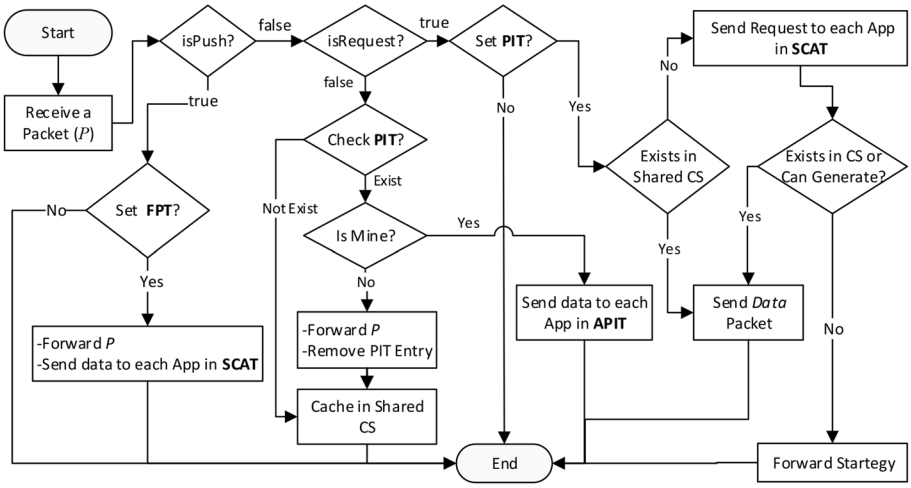

Fig. 6 The flowchart of the content retrieval mechanism for shared contents. The "set" function of PIT and FPT tables will add the new entry to the table and return Yes if the entry has been already added to the table will return No

3.4 Content Caching

Any intermediate node that receives the data packet can cache the content chunk. As it was shown in Fig. 2, in this work two types of content store are considered. According to the new model, each application has a content store (Application CS) to cache its own application-specific contents, while a shared content store (SharedCS) is considered in the network level in which the shared contents will be cached. The idea behind this modification is to support the applications with different memory requirements. Based on the new caching mechanism, each application developer can use its own caching policy on storing or removing content. Hence, each node has a SharedCS and each application has its own content store. The aggressive caching policy is used for shared contents in this work. Obviously, here application-specific contents can be cached in a vehicle only if the corresponding application is installed on the vehicle.

4 Evaluation of CCVNet

The main objective of this work was to extend the CCN model to improve the application development in a content-centric vehicular network. We have focused to overcome the challenges and make it possible to support different types of application including safety and non-safety with push and pull behaviors. In Table 2 we compared the features of CCVNet and some of the related work presented in Sect. 2.2. We have selected the works that more information was available about their architecture and the strategies used in the CCN blocks. In the upper part of the table the CCN architecture models and in the lower part the strategies and choices made in naming, caching, and content retrieval are compared. In the following of this section, the performance of CCVNet is evaluated, and the simulation results are presented.

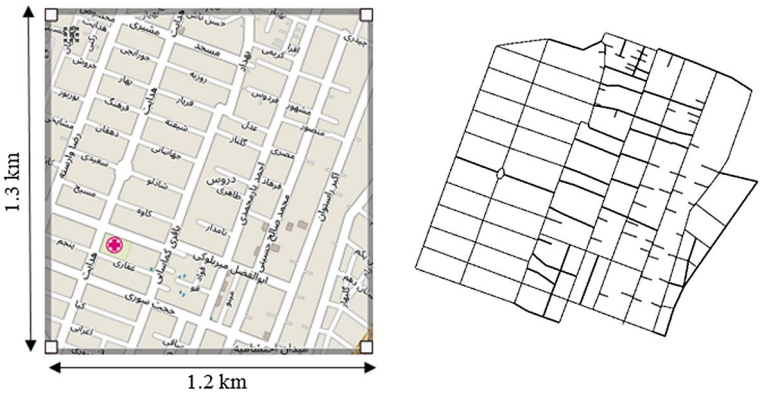

Fig. 7 The map used for vehicles mobility in the urban scenario simulation. (The OSM bounding box is: 51.4475, 35.7783, 51.4584, 35.7685)

To evaluate the performance of CCVNet in an urban scenario, SUMO [38] is used as the vehicles mobility pattern generator and NS-2.35 [39] is used for network level simulations. To generate the vehicles' mobility, we exported a part of the Tehran map from OpenStreetMap [40], which is illustrated in Fig. 7. The summary of the main simulation parameters is given in Table 3.

Three applications with different characteristics have been defined in these simulations. In addition, two types of shared content are considered.Table 4 provides the summary of these applications. In the simulations, we assume that content popularity follows a Zipf-like distribution [41]. The simulation results are averaged over ten independent runs with different seeds and reported with the 95% confidence intervals. According to the packet formats, each interest packet will be 25 bytes.

4.1 Scenario 1: Accident Occurrence

In this scenario, an accident occurrence is simulated. When the accident happens, a vehicle broadcasts a notification (one content chunk). In this simulation App1 in Table 4 is used, so we have a shared content type which is pushed to the network and 200 number of vehicles are moving all around the map. In each run, the existing vehicles are categorized based on their Euclidian distance from the accident location (50 m, 100 m, 150 m and etc.). To evaluate the performance of the presented work, two metrics are defined. For each group, the first metric is the ratio of the number of vehicles that receive the broadcasted packet to the total number of vehicles in that group. The second one is the average time between the accident occurrence and receiving the packet by vehicles; in other words, it represents the packet latency.

Figure 8 exhibits the results of the first parameter in two different situations: when the FPT table is used and when no FPT is used. In each vehicle, by storing the already forwarded pushed packet in the FPT, we avoid repetitive retransmissions of the packet and reduce the effects of the broadcast storm. As the results demonstrate, using FPT will effectively improve the content delivery to the vehicles. When an accident happens, it is very important to notify the other vehicles so that they can control their driving speed. The closer a vehicle to the accident location is, the more important receiving this notification

Table 2 A comparison of content-centric vehicular network architectures

Feature	V-NDN [26]	P-NDN [27]	CCVN [31]	RA-NDN [34]	CCVNet
Architecture					
Pull support	Yes	Yes	Yes	Yes	Yes
Push support	No	No	No	No	Yes
Provider selection	No	No	Yes	Yes	No
Shared content support	No	No	No	No	Yes
Application support	Non-safety	Non-safety	Non-safety	Non-safety	Safety + Non-safety
Naming					
Geographical scoping	Yes	Yes	No	Yes	Yes
Temporal scoping	No	No	No	Yes	Yes
Duplication detection	N/A	No	With Nonce	With nonce	With nonce
Caching					
Policy	Aggressive	N/A	N/A	Aggressive	Aggressive + application based
Type of CS	In-network central CS	In-network central CS	In-network central CS	Only on RSUs	In-network sharedCS + per application CS
Retrieval					
Controlled flooding	Furthest neighbor forwarding	Timer based	Timer based + Retransmission timeout	Furthest neighbor forwarding + direction awareness	Probabilistic forwarding

Table 3 Main simulation parameters

Parameter	Value
PHY layer	Nakagami (m = 3)
MAC layer	802.11p
Transmission range	200 m
Simulation duration	900 s
Number of vehicles	50, 100, 150, 200, 250
Vehicles maximum speed	40 km/h
Vehicles acceleration	1 m/s^2
Vehicles deceleration	5 m/s^2
Vehicles length	4 m

Table 4 Applications used in different simulation scenarios

Application	Required content types		Number of chunks	Chunk size (bytes)	Push/pull
	Shared	Description			
App 1 (accident alarm)	✓	Accident occurrence notification	1	100	Push
App2 (file sharing)	✗	Application-specific file chunks	40	1000	Pull
App3 (navigation)	✗	Application-specific map tiles	20	850	Pull
	✓	Traffic information	10	350	Pull

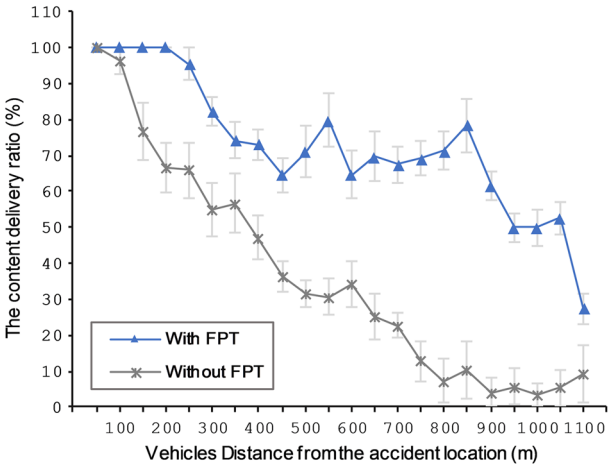

Fig. 8 The ratio of the number of vehicles that receive the packet to the total number of vehicles, based on their Euclidian distance from the accident location (App1)

becomes. According to the results by reducing the broadcast storm effect, we could improve the content delivery ratio from less than 70–100% for vehicles in the 200 m distance. It should be noted that the provided results are based on only one notification packet. In the real world by accident occurrence, the notification packets will be sent with 1 Hz frequency. Hence, the content delivery ratio will be more improved.

The results of the packet delay when FPT is used are shown in Fig. 9. According to the information provided by [7, 8] for a safety application like accident alarm, maximum

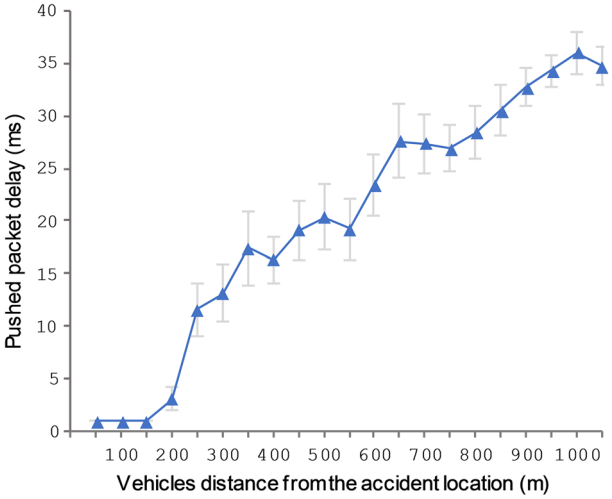

Fig. 9 The average pushed packet's delay, based on the vehicles Euclidian distance from the accident location (App1)

latency should not exceed 50 ms. According to the results shown in Figs. 8 and 9, near 100% of the vehicles that have less than 300m distance from the accident location receive the broadcasted packet under 20 ms.

4.2 Scenario 2: File Sharing Application

The performance of CCVNet is evaluated by using APP2 (in Table 4) in this scenario. App2 has one application-specific content type. In these simulations, the application is installed on 75% of the existing vehicles. At the beginning of the simulation, based on the contents' popularity a number of chunks are distributed on the vehicles that have this application. During the simulation, 30 percent of these vehicles send interests to complete their chunks. All of the vehicles can forward the interest and reply packets, but the replies are only sent by the vehicle that has App2.

4.2.1 Comparing Caching Policies

For a content-centric vehicular network, two main metrics should be evaluated. In this kind of network, the *throughput* will be defined as the ratio of the satisfied interests to the total number of sent interests. When interest is sent by a content consumer and the network delivers a copy of the requested content chunk in return, the interest is considered as satisfied (chunk-level throughput). In addition, the *content retrieval time*, the time between sending interest and receiving the corresponding content chunk, should not exceed a specific value determined according to the application requirements. As it was described in Sect. 3.4 in this work, we considered two different caching policies for application-specific contents that will be cached in the application CS:

- *CP1* The content chunk will be cached only by the application (vehicle) that has sent the interest for it;
- *CP2* All the intermediate vehicles that forward or overhear the reply packet will cache the content chunk if they have APP2.

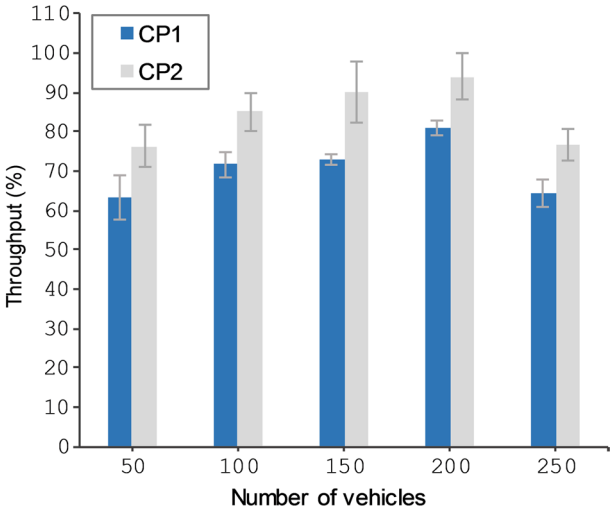

Fig. 10 Network throughput for two different cache policies (App2)

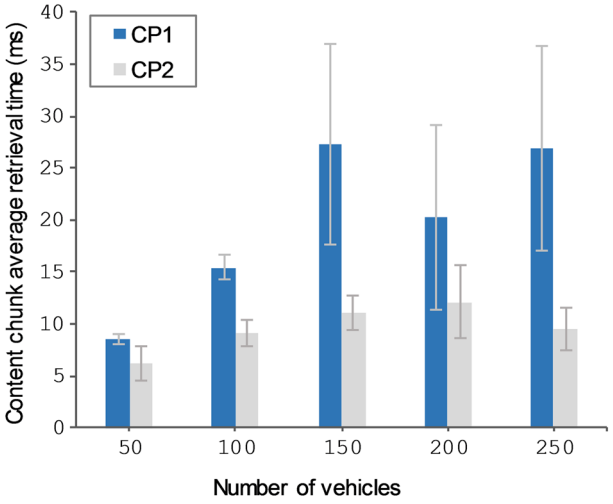

Fig. 11 Content chunk average retrieval time for two different cache policies (App2)

Figures 10 and 11 represent the simulation results of throughput and mean content retrieval time for the different number of vehicles respectively. According to the results, CP2 will lead to better results compared to CP1. Since more vehicles cache the content chunk in CP2, more vehicles will have a copy of the requested chunk and send the reply packet.

4.2.2 Forward Probability and Overhead

In the previous section, the interest forwarding mechanism was based on flooding and each vehicle that does not have the desired content chunk broadcasts the receiving interest packet. To reduce the broadcast storm effect a random variable named Forward

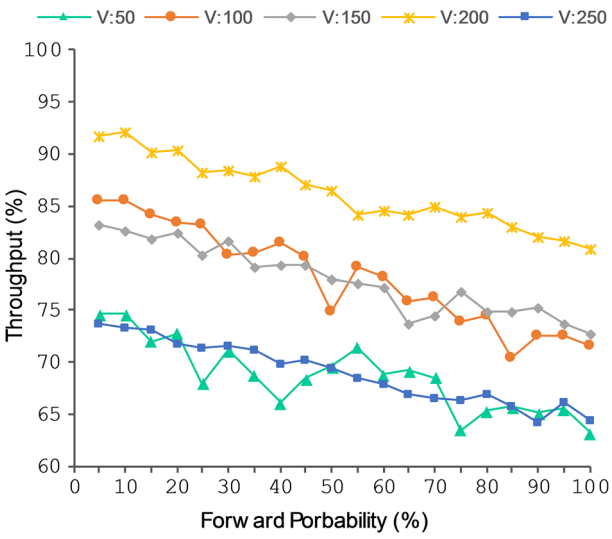

Fig. 12 Network throughput for different values of the interest forward probability, V: number of vehicles (App2)

Probability is defined. This parameter reduces the number of neighboring nodes that forward the interest packet simultaneously. The results of the network throughput for different values of the forward probability, varying from 1 to 100, are provided in Fig. 12. In these simulations, CP1 is used as the caching policy.

The results indicate that regardless of whether the network is congested or not, by reducing the number of simultaneous interest forwarding, the network performs better in content delivery. In this network interests may not be satisfied due to two different reasons: (1) no vehicle that can provide the requested content chunk exist, (2) unsuccessful interest or reply propagation, which may be caused by collisions or channel condition. In the simulation with 50 vehicles, a few numbers of vehicles have a specific content chunk, so the overall content delivery ratio (throughput) is low (reason number 1). By increasing the number of vehicles, the number of unsatisfied interests caused by the first reason will reduce. However, in the simulation with 250 vehicles, which simulates a congested network, the effect of the second reason is so high that significantly reduces the throughput.

The presented simulation shows that in a content-centric vehicular network that interest forwarding is based on flooding, the broadcast storm has a significant impact on the network performance. This impact is so much that with a simple random parameter we were able to improve the network performance. Hence, with an intelligent forward engine, we can avoid the broadcast storm problem. Improving the interest forward engine is one of our main future works.

For different interest forward probability values, the network total overhead is measured for simulations with the different number of vehicles. The overhead is calculated as: *the ratio of the total bytes of all created and forwarded interests plus the header of all created and forwarded data packets (25bytes each) to the total bytes of all data packets payload*. Due to the fact that the number of duplicated interest packets grows with the increase in the forward probability, we will have more overhead. According to the results provided by Figs. 12 and 13 with lower forward probability, the network will lead to better throughput with lower overhead.

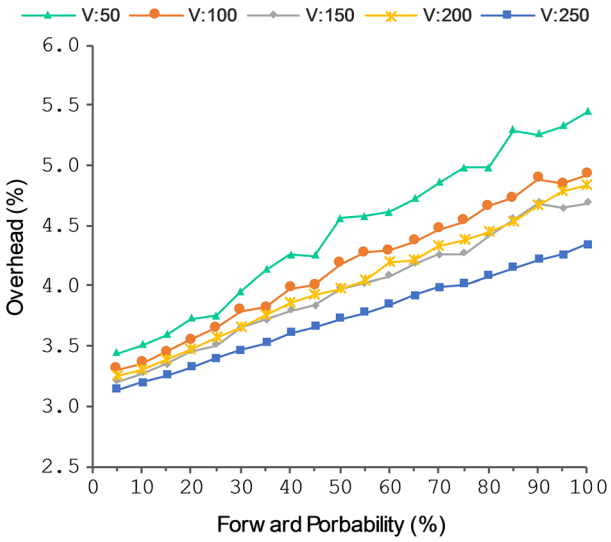

Fig. 13 Network overhead for different values of the interest forward probability, V: number of vehicles (App2)

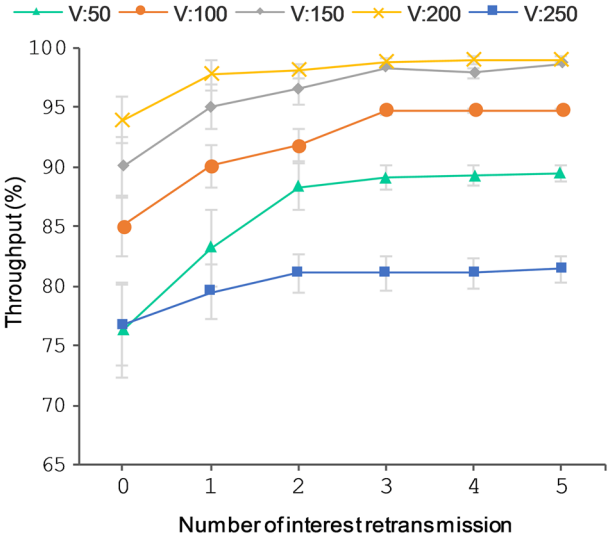

Fig. 14 Network throughput with the different number of interest retransmission, V: number of vehicles (App2)

4.2.3 Interest Retransmission

In the current set of simulations, we tried to improve the network performance by retransmission of interest packets instead of decreasing the forward probability. In these simulations, if an interest sent by the application is not satisfied, it will be retransmitted after its expiration time. An interest expiration time is set to 1 second. Figures 14 and 15 represent the simulation results of throughput and average content chunk retrieval time for the different number of vehicles respectively. The number of retransmission varies from 0 to 5. It should be noted that in these simulations, the content retrieval time is the time between the

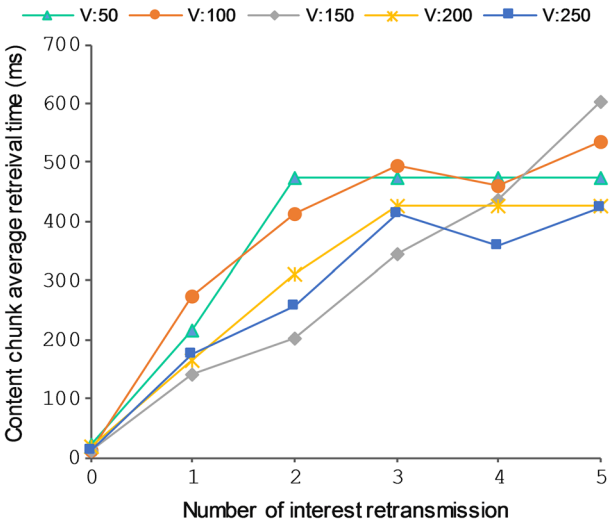

Fig. 15 Content chunk average retrieval time with different number of interest retransmission, V: number of vehicles (App2)

first sent interest and the received reply packet. It should be noted that in these simulations CP2 is considered as the caching policy.

The results in Fig. 14 indicate that in a sparse network (here v = 50) retransmission of an interest packet will effectively improve the network performance. The interest that has not been received by any neighboring vehicle and consequently not being satisfied will be finally received in second or further retransmissions. But, in a congested network (here v = 250) interest retransmission has a lower impact on network throughput. The effect of the broadcast storm problem still exists in this network. The great number of collisions caused by simultaneous broadcasts prevent interests propagation through the network. Also, according to the results, increasing the number of retransmissions up to two times will lead to better performance, but more retransmissions have no significant impact on the network throughput. According to the results in Fig. 15, although for more than two interest retransmissions, we have no considerable improvement in the network throughput, the content chunk retrieval time increases; this is caused by the total growth in the network traffic.

4.3 Scenario 3: Navigation Application

App3 (in Table 4) simulates an application in which both shared and application-specific content types are required. In the current scenario, the total number of vehicles is set to 200 and APP3 is installed on a percentage of vehicles (varying from 10 to 100%). At the beginning of the simulation, based on the contents' popularity a number of chunks are distributed on the vehicles' application CS and the shared CSs are initially empty. During the simulation, 20 vehicles (content consumer) send interest packets for both application content chunks and shared content chunks. The total number of sent interests for application-specific content chunks is 200. In addition, each of these twenty vehicles sends interests for five different geographical targets. It should be noted that, when the shared content interest is received by a vehicle, if it has APP3 and also is less than 100 meter far from the target location, will create and send a reply packet. The content retrieval mechanism is completely based on the flowcharts shown in Figs. 5 and 6 in Sect. 3.4. All the vehicles can

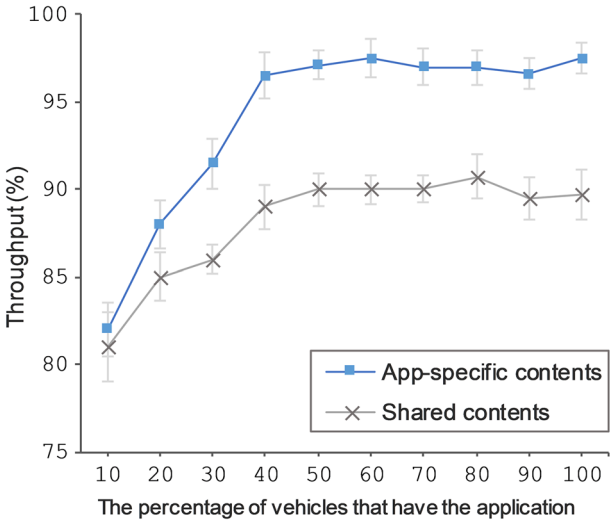

Fig. 16 Network throughput for application-specific and shared contents with different application distribution (App3)

forward the interest and reply packets, but the replies are only sent by the vehicle that has App3.

The network throughput results are shown in Fig. 16. By increasing the percentage of vehicles on which APP3 is installed up to 40%, the throughput grows. After that, increasing the number of vehicles with APP3 will have no significant impact on this metric. Furthermore, this parameter has less effect on shared content types than application-specific content types. This is caused from the fact that all 200 vehicles will cache the shared contents in their shared CS. Since a limited number of vehicles can send the reply packets for the shared content requests, the overall throughput is less than application-specific content requests.

In the second simulation, two different applications, both having App3 features are considered. Each of these applications installed on top of 40% of the vehicles. In the current scenario, all the CSs are initially empty, but we added an RSU node at the center of the map, in which three different content chunks are cached: one shared content chunk and two different application-specific chunks each for one application. During the simulation, 40 different interest packets are sent by these applications, 20 for the application specific contents and 20 for the single shared content. Figures 17 and 18 present the average result of throughput and retrieval time for both application specific and shared content chunks. As it is shown, as the time passes and more interests are sent, the network performs better for shared content retrieval than the application specific contents. The reason is that the shared content is requested with an identical name in both applications. In addition, in the proposed architecture all the vehicles in the network cache the received shared content in their SharedCs. Consequently, the shared content will faster be distributed among the vehicles on the map and we will have more content providers.

As it was described in Sect. 3.1, the idea behind adding the shared content concept to the new architecture was to allow the application designers to use their own naming scheme and also caching policies. Furthermore, there are many contents in a vehicular network that can be used by many applications. So, the proposed architecture is designed in such a way that these types of contents have a standard naming scheme and can be retrieved more efficiently.

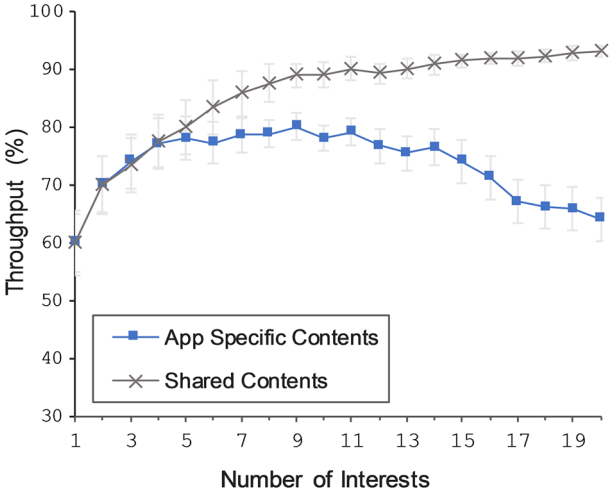

Fig. 17 Average network throughput for application-specific and shared contents, with the existence of two different applications that use the same shared content type

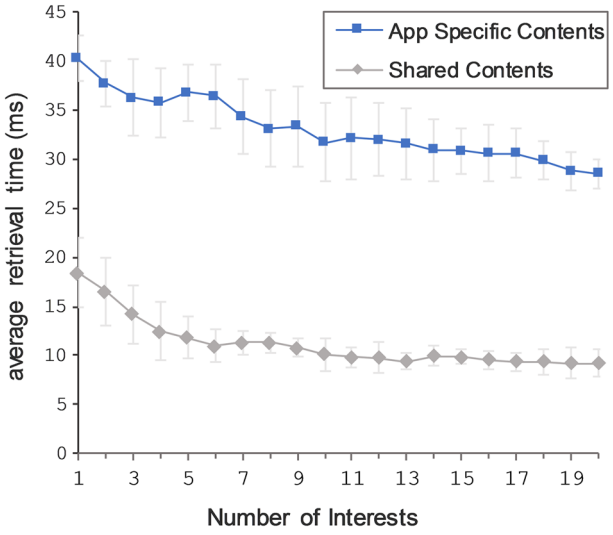

Fig. 18 Average content retrieval time (ms) for application-specific and shared contents, with the existence of two different applications that use the same shared content type

5 Conclusion

The key objective of this paper was to extend the basic CCN model to improve the application development in content-centric vehicular networks. We have focused to make it possible to support different types of application including safety and non-safety with push and pull behaviors. The contents of such a network have been divided into two groups: application-specific and shared. For each group, a naming scheme was presented that meet various applications requirements and imposed no hard restrictions for content naming on the applications. Concerning these content types, three different data structures including SCAT, APIT, and FPT have been added to the CCN model to improve the content retrieval mechanism. In CCVNet, each application has its own CS to cache application-specific contents. This feature provides applications with the ability to implement their own caching policy. On the other hand, each vehicle has

an in-network shared-CS to store shared contents. Three different scenarios have been simulated to evaluate the performance of the proposed model. The simulation results indicate CCVNet has good performance in terms of throughput, content retrieval time and overhead. For future works, we plan to consider RSUs to be involved in this content-centric vehicular network. They can be defined as the main shared content generator of the network, so the performance gain achieved with this new concept can be more investigated. By including the RSUs we will try to improve the caching policies and the content discovery and delivery mechanism. Further modifications aim to apply more intelligence to interest and data packets forwarding.

Appendix

```
Receive Packet p;
if p.isPush then
    isNewEntry = Set FPT(p);
    if isNewEntry then
        Forward(p);
        Send data to App;
    else
        drop(p);
    end
else
    if p.isRequest then
        isNewEntry = Set PIT(p);
        if isNewEntry then
            if I have App then
                if exists in cs then
                    Send(Data packet);
                    Remove PIT entry;
                else
                    Forward(p) ;
                end
            else
                Forward(p);
            end
        else
            drop(p);
        end
    else
        check PIT;
        if exists in PIT then
            if isMine then
                Send data to App;
            else
                Forward(p);
            end
            Remove PIT entry;
        else
            drop(p);
        end
    end
end
```

Algorithm 1: Application-Specific content retrieval, the pseudocode for the flowchart illustrated in Figure 5

```
Receive Packet p;
if p.isPush then
    isNewEntry = Set FPT(p);
    if isNewEntry then
        foreach App in SCAT do
            Send data to App;
        end
        Forward(p);
    else
        drop(p);
    end
else
    if p.isRequest then
        isNewEntry = Set PIT(p);
        if isNewEntry then
            if exists in SharedCS then
                Send(Data packet);
            else
                foreach App in SCAT do
                    Send Request to App;
                    if exists in CS OR Can generate then
                        Send(Data packet);
                        Remove PIT entry;
                    else
                        Forward(p);
                    end
                end
            end
        else
            drop(p);
        end
    else
        check PIT;
        if exists in PIT then
            if isMine then
                foreach App in APIT do
                    Send data to App;
                end
            else
                Forward(p);
            end
            Remove PIT entry;
        else
            Cache data in SharedCS;
            drop(p);
        end
    end
end
```

Algorithm 2: Shared content retrieval, the pseudocode for the flowchart illustrated in Figure 6

References

1. Lu, N., Cheng, N., Zhang, N., Shen, X., & Mark, J. W. (2014). Connected vehicles: Solutions and challenges. *IEEE Internet of Things Journal*, *1*(4), 289–299. https://doi.org/10.1109/JIOT.2014.2327587.
2. Campolo, C., Molinaro, A., & Scopigno, R. (2015). From today' VANETs to tomorrow's planning and the bets for the day after. *Vehicular Communications*, *2*(3), 158–171. https://doi.org/10.1016/j.vehcom.2015.06.002.
3. Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, *54*(15), 2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010.
4. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generat Comput Syst*, *29*(7), 1645–1660. https://doi.org/10.1016/j.future.2013.01.010.
5. Zanella, a, Bui, N., Castellani, a, Vangelista, L., & Zorzi, M. (2014). Internet of Things for smart cities. *IEEE Internet of Things Journal*, *1*(1), 22–32. https://doi.org/10.1109/JIOT.2014.2306328.
6. Altintas, O., Dressler, F., & Hagenauer, F. (2015). Making Cars a Main ICT resource in smart cities. In *IEEE conference on computer communications workshops (INFOCOM WKSHPS)* (pp. 582–587). https://doi.org/10.1109/INFCOMW.2015.7179448.
7. Zheng, K., Zheng, Q., Chatzimisios, P., Xiang, W., & Zhou, Y. (2015). Heterogeneous vehicular networking: A survey on architecture, challenges and solutions. *IEEE Communications Surveys & Tutorials*, *17*(4), 2377–2396. https://doi.org/10.1109/COMST.2015.2440103.
8. Hameed Mir, Z., & Filali, F. (2014). LTE and IEEE 802.11p for vehicular networking: a performance evaluation. *EURASIP Journal on Wireless Communications and Networking*, *2014*(1), 89. https://doi.org/10.1186/1687-1499-2014-89.
9. Felemban, E., & Sheikh, A. (2014). A review on mobile and sensor networks innovations in intelligent transportation systems. *Journal of Transportation Technologies*, *4*, 196–204. https://doi.org/10.4236/jtts.2014.43020.
10. Amadeo, M., Campolo, C., & Molinaro, A. (2016). Information-centric networking for connected vehicles: A survey and future perspectives. *IEEE Communications Magazine*, *54*(2), 98–104. https://doi.org/10.1109/MCOM.2016.7402268.
11. Araniti, G., Campolo, C., Condoluci, M., Iera, A., & Molinaro, A. (2013). LTE for vehicular networking: A survey. *IEEE Communications Magazine*, *51*(5), 148–157. https://doi.org/10.1109/MCOM.2013.6515060.
12. Festag, A. (2015). Standards for vehicular communication—From IEEE 802.11p to 5G. *e & i Elektrotechnik und Informationstechnik*, *132*(7), 409–416. https://doi.org/10.1007/s00502-015-0343-0.
13. DSRC: The future of safer driving. Retrieved April 2018, fromhttp://www.its.dot.gov/factsheets/dsrc_factsheet.htm.
14. Abboud, K., Omar, H. A., & Zhuang, W. (2016). Interworking of DSRC and cellular network technologies for V2X communications: A survey. *IEEE Transactions on Vehicular Technology*, *65*(12), 9457–9470. https://doi.org/10.1109/TVT.2016.2591558.
15. Amadeo, M., Campolo, C., Molinaro, A., & Ruggeri, G. (2014). Content-centric wireless networking: A survey. *Computer Networks*, *72*, 1–13. https://doi.org/10.1016/j.comnet.2014.07.003.
16. Talebifard, P., Leung, V. C. M., Amadeo, M., Campolo, C., & Molinaro, A. (2015). Information-centric networking for VANETs. *Vehicular ad hoc Networks* (pp. 503–524). New York: Springer. https://doi.org/10.1007/978-3-319-15497-8.
17. Yu, Y. T. (2014). Information-centric vehicular ad-hoc networks: Challenges and solutions, Ph.D. dissertation, University of California, Los Angeles, CA.
18. Bouk, S., Ahmed, S., & Kim, D. (2015). Vehicular content-centric network (VCCN): A survey and research challenges. In *Proceedings of the 30th annual ACM symposium on applied computing* (pp. 695–700). https://doi.org/10.1145/2695664.2695844.
19. Wang, J., Wakikawa, R., & Zhang, L. (2010). DMND: Collecting data from mobiles using Named Data. *IEEE Vehicular Networking Conference*, *2010*, 49–56. https://doi.org/10.1109/VNC.2010.5698270.
20. Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., & Braynard, R. (2009). Networking named content. In *Proceedings of the 5Th international conference on emerging networking experiments and technologies* (pp. 1–12). https://doi.org/10.1145/1658939.1658941.
21. Amadeo, M., Campolo, C., Quevedo, J., Corujo, D., Molinaro, A., Iera, A., et al. (2016). Information-centric networking for the internet of things: Challenges and opportunities. *IEEE Network*, *30*(2), 92–100. https://doi.org/10.1109/MNET.2016.7437030.

22. Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., & Ohlman, B. (2012). A survey of information—Centric networking. *IEEE Communications Magazine*, *50*(7), 26–36. https://doi.org/10.1109/MCOM.2012.6231276.
23. Saxena, D., Raychoudhury, V., Suri, N., Becker, C., & Cao, J. (2016). Named data networking: A survey. *Computer Science Review*, *19*, 15–55. https://doi.org/10.1016/j.cosrev.2016.01.001.
24. Nicanfar, H., TalebiFard, P., Zhu, C., & Leung, V. (2013). Efficient security solution for information-centric networking. In *2013 IEEE international conference on green computing and communications and IEEE Internet Of Things and IEEE cyber, physical and social computing* (pp. 1290–1295). https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.224.
25. Wang, L., Wakikawa, R., Kuntz, R., Vuyyuru, R., & Zhang, L. (2012). Data naming in Vehicle-to-Vehicle communications. In *2012 proceedings IEEE INFOCOM workshops* (pp. 328–333). IEEE. https://doi.org/10.1109/INFCOMW.2012.6193515.
26. Grassi, G., Pesavento, D., Pau, G., Vuyyuru, R., Wakikawa, R., & Zhang, L. (2014). VANET via named data networking. In *2014 IEEE conference on computer communications workshops (INFOCOM WKSHPS)* (Vol. 1, pp. 410–415). https://doi.org/10.1109/INFCOMW.2014.6849267.
27. Amadeo, M., Campolo, C., & Molinaro, A. (2016). Named data networking for priority-based content dissemination in VANETs. In *IEEE international symposium on personal, indoor and mobile radio communications, PIMRC* (pp. 1–6). IEEE. https://doi.org/10.1109/PIMRC.2016.7794616.
28. Tonguz, O., Wisitpongphan, N., Parikh, J., Bai, F., Mudalige, P., & Sadekar, V. (2006). On the broadcast storm problem in ad-hoc wireless networks. In *2006 3Rd international conference on broadband communications, networks and systems* (pp. 1–11). https://doi.org/10.1109/BROADNETS.2006.4374403.
29. Ahmed, S. H., Bouk, S. H., & Kim, D. (2015). RUFS: RobUst forwarder selection in vehicular content-centric networks. *IEEE Communications Letters*, *19*(9), 1616–1619. https://doi.org/10.1109/LCOMM.2015.2451647.
30. Wang, L., Afanasyev, A., & Kuntz, R. (2012). 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* (Vol. 12, pp. 7–12). https://doi.org/10.1145/2248361.2248365.
31. Amadeo, M., Campolo, C., & Molinaro, A. (2013). Enhancing content-centric networking for vehicular environments. *Computer Networks*, *57*(16), 3222–3234. https://doi.org/10.1016/j.comnet.2013.07.005.
32. Arnould, G., Khadraoui, D., & Habbas, Z. (2011). A self-organizing content centric network model for hybrid vehicular ad-hoc networks. In *Proceedings of the first ACM international symposium on Design and analysis of intelligent vehicular networks and applications—DIVANet '11* (p. 16-22). New York, New York, USA: ACM Press. https://doi.org/10.1145/2069000.2069004.
33. Yu, Y.-T., & Gerla, M. (2016). Information-centric VANETs: A study of content routing design alternatives. In *2016 international conference on computing, networking and communications (ICNC)* (pp. 1–5). IEEE. https://doi.org/10.1109/ICCNC.2016.7440705.
34. Tiennoy, S., & Saivichit, C. (2018). Using a distributed roadside unit for the data dissemination protocol in VANET with the named data architecture. *IEEE Access*, *6*, 32612–32623. https://doi.org/10.1109/ACCESS.2018.2840088.
35. Amadeo, M., Campolo, C., & Molinaro, A. (2012). CRoWN: Content-centric networking in vehicular ad hoc networks. *IEEE Communications Letters*, *16*(9), 1380–1383. https://doi.org/10.1109/LCOMM.2012.072012.120282.
36. Majeed, M. F., Ahmed, S. H., & Dailey, M. N. (2017). Enabling push-based critical data forwarding in vehicular named data networks. *IEEE Communications Letters*, *21*(4), 873–876. https://doi.org/10.1109/LCOMM.2016.2642194.
37. ICN Research Challenges, draft-irtf-icnrg-challenges-06. Retrieved September 2018, from https://tools.ietf.org/html/draft-irtf-icnrg-challenges-06.
38. SUMO—Simulation of Urban MObility. Retrieved 25 September 2018, from www.sumo.dlr.de.
39. The Network Simulator - ns-2. Retrieved 25 September 2018, from http://www.isi.edu/nsnam/ns/.
40. OpenStreetMap. Retrieved 19 December 2018, from https://www.openstreetmap.org.
41. Breslau, L., Cao, Pei, Fan, Li, Phillips, G., & Shenker, S. (1999). Web caching and Zipf-like distributions: Evidence and implications. In *IEEE INFOCOM '99, conference on computer communications, proceedings, eighteenth annual joint conference of the IEEE computer and communications societies*. The Future Is Now (Cat. No.99CH36320) (pp. 126–134). https://doi.org/10.1109/INFCOM.1999.749260.

Rojin Tizvar received her B.S. and M.S. degrees in Computer Engineering from Shahid Beheshti University, Tehran, Iran, in 2009 and 2012 respectively. She is currently a Ph.D. student at Shahid Beheshti University and works as a researcher at the Computer networks and Security Laboratory (CNSLab). Her main fields of interest include Internet of Things, Vehicular Networks, Intelligent Transportation Systems, Content Centric Networking, Wireless Sensor Networks and Cognitive Radio Networks.

Maghsoud Abbaspour received the B.Sc., M.S., and Ph.D. degrees in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 1992, 1995 and 2003, respectively. He joined Faculty of Computer Science and Engineering, Shahid Beheshti University since 2005 as an assistant professor. He now is an associate professor and is head of Wireless Networks and Network Security Laboratory in Shahid Beheshti University. He is interested in Wireless Sensor Networks, peer to peer networks, Internet of things, malware detection, and network security.