World Academy of Science, Engineering and Technology
International Journal of Electronics and Communication Engineering
Vol:17, No:8, 2023

SPA-VNDN: Enhanced Smart Parking Application by Vehicular Named Data Networking

Bassma Aldahlan, Zongming Fei

Abstract—Recently, there is a great interest in smart parking application. Theses applications are enhanced by a vehicular ad-hoc network, which helps drivers find and reserve satiable packing spaces for a period of time ahead of time. Named Data Networking (NDN) is a future Internet architecture that benefits vehicular ad-hoc networks because of its clean-slate design and pure communication model. In this paper, we proposed an NDN-based frame-work for smart parking that involved a fog computing architecture. The proposed application had two main directions: First, we allowed drivers to query the number of parking spaces in a particular parking lot. Second, we introduced a technique that enabled drivers to make intelligent reservations before their arrival time. We also introduced a "push-based" model supporting the NDN-based framework for smart parking applications. To evaluate the proposed solution's performance, we analyzed the function for finding parking lots with available parking spaces and the function for reserving a parking space. Our system showed high performance results in terms of response time and push overhead. The proposed reservation application performed better than the baseline approach.

Keywords—Cloud Computing, Vehicular Named Data Networking, Smart Parking Applications, Fog Computing.

I. Introduction

SMART parking applications have attracted many researchers in recent years, especially with the increasing number of vehicles in urban cities. Finding an available parking spot could be stressful in big crowded cities. Drivers may circle around several times to find parking spaces, making it time-consuming and a waste of gasoline. When more than one vehicle attempts to find parking space in a mostly vacant parking area, the traffic congestion and pollution could increase. To solve this issue, we need a smart parking application to help us find a suitable parking lot.

Some projects such as SFpark (San Francisco) and LA Express Park (Los Angeles) [1] take advantage of smart parking applications using smartphones. In these projects, many sensors were deployed to collect the sensed data and send them to the nearby meters. Drivers with smartphone devices can request the availability of parking spaces. These projects are promising solutions. However, they need improvements on several aspects to satisfy the demand in real-time. First, all drivers can get the same information regarding parking space availability. Second, real-time availability data are only available for vehicles near the parking location. Third, installing sensors should be more robust to avoid road obstacles; interpreting and processing those sensed data should be done through a wireless network to prevent sending high traffic to the same spectrum.

Intuitively, the development of a smart parking solution can reduce parking search time and traffic congestion. It can also reduce environmental pollution and fuel consumption. In addition, smart parking brings economic benefits by increasing activities and business opportunities. In this context, a vehicular ad-hoc network (VANET) is a critical technology that can be considered in smart parking applications to allow vehicles to communicate with each other or enable infrastructure, such as road side units (RSUs) and base stations (BSs), to serve the smart parking systems. In VANETs, the main communication modes are vehicle-to-vehicle (V2V), in which a VANET can form a pure communications mode between on-board units (OBUs) via a wireless network (IEEE 802.11p), and vehicle-to-infrastructure (V2I), the communication between vehicles and infrastructure, such as RSUs and BSs through either IEEE 802.11p or cellular communication.

Using V2V communication in smart parking systems could face challenges. In VENETs, vehicles are limited in sharing their sensed information via V2V communication. Also, sending huge information through relay nodes can be energy-consuming. On the other hand, VANET supports real-time information dissemination. By utilizing V2I communication, the smart parking system performs better. However, deploying RSUs everywhere can be costly in terms of installation, deployment, and maintenance. RSUs' capability in computation and storage can be limited as well.

Currently, smart parking applications use an IP-based network to enable communication. However, an IP-based network has many issues: First, its limited resources do not make it easy to assign a unique IP address to all sensors and vehicles. Second, achieving requests via those unique addresses in a mobile environment like VANETs is not trivial. Third, the IP-based method depends on where the data reside, not on where the content is, forcing us to find the target node that has the information regarding the smart parking system [2].

Vehicular named data networking (V-NDN) has brought more attention to academic and industrial fields. It replaced the traditional IP network with NDN to overcome the limitations in IP networks. V-NDN is a promising network architecture that supports high mobility and intermittent disconnectivity. The nature of NDN, where the data content can be fetched from the producers or any associated cache, can play a role in VANETs, such as by enhancing self-certification and layer security. NDN also supports multicast communication in a

B. Aldahlan is with Yanbu University College, Yanbu, Saudi Arabia (correspondence author, e-mail: aldahlanb@rcyci.edu.sa, baldahlan@uky.edu).
Z. Fei is with University of Kentucky, Lexington, KY 40503 USA. He is now with the Department of Computer Science (e-mail: fei@netlab.uky.edu).

mobile environment. However, it still has challenges that need to be addressed.

This paper provides a design of an NDN-based framework for smart parking applications, taking advantages of the benefit of fog-computing to answer drivers' queries about whether there is parking space available in a particular geographical area. Further, these applications introduce a technique that allows drivers to make reservations ahead of time. We assume that sensors are installed in the parking lots to collect availability information, and sensors on vehicles can help with the parking application. We also introduced a "push-based model" to integrate sensor nodes into the system for sensor nodes to deliver the information proactively.

II. Related Work

Recently, smart parking applications have been considered as important applications in VANET and have attracted many researchers in the industrial and research fields. Many survey studies [1], [3]–[5] were conducted to investigate these applications. Lin et al. [1] proposed a survey paper that depends on a smart parking ecosystem and a comprehensive classification of smart parking applications by identifying the functionalities of the existing work and their problematic priorities. They divided the existing work into three macro-themes based on the target of each work: information collection tools that assist in gathering data, such as sensors, actuators, and parking meters; system deployment, which handles the software system exploitation that assists with the statistical analysis of the collected data and provides data prediction; service dissemination, which deals with the relationship between the gathered information and some social feature related to the driver's behaviors, including parking competition, information dissemination, and packing behavior. Both [4] and [3] discussed some technologies about parking availability monitoring, parking reservation, and dynamic pricing. In [4], the authors investigated the different technologies regarding parking availability monitoring. These technologies help in the dissemination of the associated information regarding parking space availability and parking reservation. The authors aimed to ensure higher customer satisfaction and increase revenue from parking services. Parking reservation allows the drivers to reserve the parking space in advance before arriving at the parking area. The authors studied parking infrastructure from different perspectives. Faheem et al. [5] presented a review paper showing different intelligent parking services (IPS), which are used for parking guidance and parking facility management. The authors discussed all the techniques that contribute to having an efficient and modern parking system. Faheem et al. [5] provided an economic analysis that assessed the project's feasibility.

Parking reservation enables drivers to reserve the parking spaces before their arrival time. Integrating reservation policies with smart parking provides many benefits for both the drivers and the managers of parking areas. Delot et al. [6] proposed a reservation protocol that allows vehicles to search for available parking slots based on their requested event. Vehicles with similar interests in a relevant event will be cooperatively gathered. Besides that, the relevant direction of vehicles is taken into account to support the reservation strategy. They aimed to avoid competition between the vehicles. Doulamis et al. [7] introduced an intelligent parking reservation management that relies on optimal strategy, aiming to promote the service quality of drivers and increase the parking spaces' utilization. The proposed approach utilizes the interval scheduling principles represented as a list of parking requests provided as a set of requested time intervals. The authors presented a scheduler that can determine whether to accept the task and assign it to some resource or to deny it. They also introduced an adaptive pricing policy proportional to the rejection probability of a parking request.

Karbab et al. [8] proposed a scalable and low-cost car parking framework (CPF). The authors introduced driver guidance, automatic payment, parking lot retrieval, security, and vandalism detection. They also introduced a standard I2C protocol to cluster a group of sensors into a single mote. They used smartphone applications to reserve parking slots and hybrid wireless communications. The GPS helps in getting the real-time location and guidance toward the destination. However, smartphone applications are only beneficial if the driver is within 2 km of the location.

Smart parking applications involve a high amount of data traffic through various sensors. This high amount of traffic caused by the Internet of Things (IoT) sensed data are considered. Both [9] and [10] have studied the ability of cloud computing, especially fog computing, on VANET, to support the computational demands and reduce the response time. Fog computing at edges can bring the benefits of computing, processing, and storage to the edges to be more closer to the sensor devices. Hou et al. [11] introduced the idea of exploiting fog computing in VANET. Vehicular fog computing (VFC) utilizes a vehicular node to act as a fog node for communication and computation. The authors used the mobile and parked vehicles as infrastructure to perform communication and computation. Xiao and Zhu [12] also introduced an idea similar to VFC. They considered the connected vehicles as mobile fog nodes. They presented cost-effective and on-demand fog computing for vehicular applications. Tang et al. [13] proposed a parking slot allocation strategy that considered real-time parking slot information by exploiting the fog computing-based capabilities on a smart parking architecture, aiming to enhance smart parking in real-time. The deployed fog nodes (RSUs) at parking lots can communicate and cooperate, allowing parking request processing. Meanwhile, the centralized cloud can promote smart parking capability by enforcing global optimization on parking requests. They provided an allocation strategy that considered the comprehensive factors that can affect decision making, such as walking, driving, and waiting costs. Installing, deploying, and maintaining an RSU for every parking to achieve one-hop communication could be expensive. Yi et al. [14] proposed a packing reservation auction system that uses cloud fog computing associated with parked vehicle cloud. They aimed to guide mobile vehicles to the available parking spaces with less effort. They also used the fog capability of

parked vehicles to help the delay-sensitive computing services through monetary rewards to compensate for their service cost and allocate the workload to each CPU. This solution results in performance due to the enhancement of the fog node controller, the smart vehicles, and the parking places. However, this work considered only the parked vehicles located in some parking areas and did not consider on-street parking. The authors did not mention any criteria to select those parked vehicles.

Despite the previous solutions, the smart parking application is still in its early stages and needs improvements. In this study, we developed a VNDN-based smart parking application using a fog computing architecture that supports IoT data collection and avoids the overcrowdedness of the central cloud approach.

III. SPA-VNDN: Design

A smart parking application is developed to provide two functions. One is to allow a driver to find parking lots that have available parking spaces. The driver can send out the request as an interest packet in the NDN architecture and the replies can come from multiple parking lots. It is up to the driver to pick any specific parking lot. Normally, the driver wants to find available parking space at this moment. We will discuss a case in which the driver may be interested in the availability in the future. This is related to the function we will discuss next.

The other function is to allow a driver to reserve a parking spot in a particular parking lot for a period of time. This can happen after the driver finds a list of parking lots with available spaces in some future time. Then, the driver picks one to make a reservation with the arrival and departure times specified. Similar to the previous function, the driver can also provide geographical-area information in the request packet so that it can be forwarded to the correct location, rather than using the broadcast method.

The smart parking application adopts a fog cloud architecture to provide a flexible structure and improve the response time. The NDN naming scheme was designed to identify the request, either for finding an available parking space or making a reservation for the future. We introduced a push-based model into the NDN architecture to streamline the process for allowing sensor nodes installed to get the information into the system.

A. Design Assumption of the Smart Parking Application

We consider a topology with roads with traffic in both directions and interactions where decisions are made to change directions. In addition, sensor nodes will be installed to collect real-time information, and cloud computing elements will be used to meet the communication and computing requirements of the application. Fig. 1 illustrates the main components.

1) *Vehicles*: Vehicles can communicate with other vehicles and with infrastructure units (fog nodes). They are equipped with a Global Positioning System (GPS), which can tell the location of the vehicle. We can classify vehicles into two types: static vehicles (parked vehicles) that are parked in a parking garage or the curbside of roads, and dynamic vehicles (mobile vehicles) that are moving.
2) *Sensors*: Sensors are installed to monitor a variety of status. In particular, the parking application may have sensors to monitor whether a parking spot is free. In addition, modern vehicles are equipped with many sensors that can also contribute information to the parking application, such as location, speed, and direction.
3) *Road Side Units*: RSUs are computing, storage, and communication resources installed on the road side for supporting modern transportation systems. They are an important component of VANET and will be a good candidate as the fog node in the fog cloud architecture. However, not all roads have RSUs installed. Vehicles and RSUs communicate through WiFi IEEE802.11p/WAVE technology, and the quality of the communication can be significantly better than vehicle to vehicle communication.
4) *Dedicated Servers*: Dedicated servers may be installed in big parking garages for managing the parking space. They can also be incorporated into the fog cloud architecture.
5) *Central Cloud*: Central cloud provides a central storage and processing service for the fog nodes connected to the architecture. It provides global information necessary for coordination among the fog nodes.

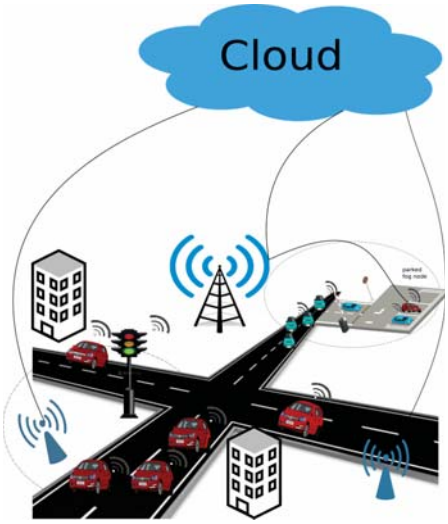

Fig. 1 Network Topology

1) Fog Cloud Architecture: We considered several architectural candidates for the smart parking application. One possibility is that we can depend on RSUs and dedicated servers on larger parking garage to handle the requests, without any help from the cloud architecture. There are four limitations. First, there are smaller parking lots without dedicated servers or not covered by RSUs. We will not be able to include those parking spaces into the application. Second, if the dedicated servers or the RSUs crashes, the information stored will be lost. For example, the reservation information will not be able to be recovered and agreements

with customers will not be able to be honored. Third, different parking lots will not be able to cooperate with each other, such as recommending alternative parking places if the current lot is full. And finally, deploying RSUs and dedicated servers can be costly.

The second possibility is that we can use a centralized cloud server to handle all the requests from users. There are two potential issues. First, the central cloud becomes the concentration point of all requests. It can take a longer time for a user to get the response than the case in which a request can be processed by a distributed and close-by server. Second, the state information collected by the sensors will be sent over to the cloud. This can be huge, depending on the scope of parking spaces the central cloud is handling.

Instead, we consider an architecture based on the fog cloud computing. We still have a central cloud service that has a large pool of processing and storage resources that can be allocated on demand to meet the ever-changing requirement of the system. In addition, we have fog nodes distributed in different parts of the system, located at the edge of cloud and close to the parking space they serve. They will handle the request for the spaces they are in charge. The central cloud server also acts as the backup service in case of some critical states at distributed fog nodes are lost. It also has the authority to manage the fog infrastructure by authorizing the setup and tear-down of fog nodes. The fog nodes can be RSUs, dedicated servers, or even vehicles. Usually, status information generated by the sensor nodes is sent to the fog nodes. Only aggregated information or critical states will be forwarded to the central cloud. For example, whether a particular parking spot is occupied is kept at the fog node, while the total number of free spaces may be forwarded from the fog node to the central cloud server.

For large parking garages, dedicated servers will be the first choice as the fog node, because their processing units are more powerful and storage space is larger. Similarly, when there is an RSU next to a parking lot, we will prefer using the RSU as a fog node to take care of processing requirements. These two kinds of fog nodes are easy to set up and we will not focus on them in our following discussion. However, for parking lots that do have dedicated servers available or are not covered by RSUs, or they are out of service due to hardware or software failure. We proposed to utilize the capability of Vehicular Fog Computing (VFC) [11], [15], [16] to set up vehicle-based fog nodes to handle the requests for these lots. The VFC is especially useful for handling the cases such as curbside parking and small parking areas.

We have to address several issues. First, we need to select a vehicle as a fog node. Generally, we prefer to select a parked vehicle because it will stay in place for a longer period of time. Second, we need to enable vehicles and sensor devices to communicate with the selected fog node. The fog node maintains computing, storage and communication capabilities to handle users requests. Third, we need to connect the fog node with the central could so that it can be authorized to make decisions and be trusted by sensor nodes and other vehicles. Also we need to handle the transition to other vehicles once the current fog node needs to leave the location.

B. Fog Cloud Architecture

We considered several architectural candidates for the smart parking application. One possibility is that we can depend on RSUs and dedicated servers on a larger parking garage to handle the requests, without any help from the cloud architecture. There are four limitations. First, there are smaller parking lots without dedicated servers or not covered by RSUs. We will not be able to include those parking spaces into the application. Second, if the dedicated servers or the RSUs crash, the information stored there will be lost. For example, the reservation information will not be recovered, and agreements with customers will not be honored. Third, different parking lots will not be cooperate with each other, such as by recommending alternative parking places if the current lot is full. Finally, deploying RSUs and dedicated servers can be costly.

The second possibility is that we can use a centralized cloud server to handle all the requests from users. There are two potential issues. First, the central cloud becomes the concentration point of all requests. It can take longer for a user to get the response than when a request is processed by a distributed and close-by server. Second, the state information collected by the sensors will be sent over to the cloud. This can be huge, depending on the scope of parking spaces the central cloud is handling.

Instead, we consider an architecture based on fog cloud computing. We still have a central cloud service with a large pool of processing and storage resources that can be allocated on demand to meet the ever-changing requirement of the system. In addition, we have fog nodes distributed in different parts of the system, located at the edge of cloud and close to the parking space they serve. They will handle the request for the spaces they are in charge of. The central cloud server also acts as the backup service in case of some critical states at distributed fog nodes are lost. It also has the authority to manage the fog infrastructure by authorizing the setup and tear-down of fog nodes. The fog nodes can be RSUs, dedicated servers, or even vehicles. Usually, status information generated by the sensor nodes is sent to the fog nodes. Only aggregated information or critical states will be forwarded to the central cloud. For example, whether a particular parking spot is occupied is kept at the fog node, while the total number of free spaces may be forwarded from the fog node to the central cloud server.

For large parking garages, dedicated servers will be the first choice as the fog node, because their processing units are more powerful and storage space is larger. Similarly, when there is an RSU next to a parking lot, we will prefer using the RSU as a fog node to take care of processing requirements. These two kinds of fog nodes are easy to set up, and we will not focus on them in our following discussion. However, for parking lots that are not dedicated servers, are not covered by RSUs, or are out of service due to hardware or software failure, we proposed to utilize the capability of VFC [11], [15], [16] to set up vehicle-based fog nodes to handle the requests for these lots. The VFC is especially useful for handling cases, such as curbside parking and small parking areas.

We have to address several issues. First, we must select a vehicle as a fog node. Generally, we prefer to select a parked vehicle because it will stay in place for a longer period of time. Second, we must enable vehicles and sensor devices to communicate with the selected fog node. The fog node maintains computing, storage, and communication capabilities to handle user requests. Third, we should connect the fog node with the central could so that it can be authorized to make decisions and be trusted by sensor nodes and other vehicles. Also, we need to handle the transition to other vehicles once the current fog node leaves the location.

1) Selection of Fog Nodes: Ultimately, the central cloud has the authority to select which vehicle can be the fog node for a particular parking space. Information about potential candidates – parked vehicles – should be obtained by the central cloud to make the decision. With no existing fog node, interested parked vehicles can send information to the cloud directly. This may be done by vehicle to vehicle communications until the message reaches an RSU or other fog node handling a different parking area that has a channel to the central cloud. With an existing fog node in the area, information about parked vehicles can be collected and sent to the central cloud so that a new fog can be selected once the current fog node leaves the area.

The attributes of the parked vehicles that will be sent to the central cloud include the following:

1) Vehicle ID: the identification of the vehicle.
2) Current location: the x and y coordinates of the vehicle.
3) Storage size, computing and communication capacities: the processing power of the vehicle.
4) Parking duration: the period that the vehicle will be parked.
5) Willingness: a vehicle is willing to participate in forming a vehicular cloud. A vehicle sending information usually has a high level of willingness to participate.

The central cloud maintains a data repository to gather vehicles and the fog cloud information. Based on the data collected regarding the vehicle's information, the central cloud uses some criteria to select the fog node as the initial setup for a parking space or as the replacement for the current fog node that will leave the area. First, the central cloud has a set of thresholds, and only those nodes with metrics equal to or greater than the thresholds can become candidates. The criteria are as follows:

1) Computing, storage and communication capacities: These vary from vehicle to vehicle. The central cloud specifies a threshold and filters out all vehicles that do not meet the threshold requirement.
2) Duration of being active: Another factor to consider is how long the vehicle can provide service. The central cloud will balance the capacities and duration of service.
3) Behavior rate: In addition to the vehicles' willingness to serve, the central cloud also maintains information about their past behavior. Vehicles with high ratings will be preferred. The central cloud performs an adaptive adjustment to rate the vehicles' node behavior based on the previous monitoring of the participating vehicles. Each time a vehicle participates as a fog node, the cloud monitors the fog nodes and rates them based on how long they serve and based on the other vehicles' performance.

2) Communication with Fog Nodes: After a static vehicle is selected as the fog node, it will broadcast an *advertisement* message to inform other nodes in the same geo-area, including sensor devices and other vehicles.

The following naming convention is used for the advertisement message: "/advertisement/Fog_nodeID/ current position/type/duration." In this format, "advertisement" indicates the type of message, which is a broadcast message; "Fog_nodeID" represents the identification of the selected fog node; "current position" indicates the current position of the fog node at the time; "type" indicates whether the fog node is an RSU, a dedicated server or a parked vehicles; and "duration" shows the duration in which the fog node will be active.

After receiving this message, other nodes will receive information about the fog node, including Fog_nodeID, current position, type, and duration. The device (e.g., vehicle or sensor) can include the interest name of the advertisement message in its PIT table. Intermediate nodes in the NDN architecture also record it as a pending interest with the incoming interface.

When the PIT entry is set up in the PIT table of a vehicle or a sensor node, it can push any sensed data messages or send any other requests regarding the smart parking applications to the fog node. The difference is that the device can periodically send updated state information, not just one message, to meet the requirement of the interest packet. The device can identify the message as push-based status information associated with the fog node using its "Fog_nodeID." To support the push-based model, we must modify the PIT table entry at intermediate nodes. Instead of removing the entry after one message meeting, the interest is forwarded via the incoming interfaces. It will keep the entry for the duration specified in the advertisement message sent by the fog node.

C. Finding Parking Lots with Available Parking Space

The first function we want to implement for the smart parking application is to enable drivers to find parking lots with available parking space. We assume that drivers send a query to check parking space availability for a specific area. When these interest packets arrive in the target area, all parking lots with available parking space will reply.

To realize the function, we assume that sensors (IoT devices) are installed to monitor the parking space in each parking lot. We assume that there are n number of parking spots denoted by $P = \{p_1, p_2, ..., p_n\}$. The real-time information of a parking spot can be collected by a parking sensor, and it will be pushed to the associated fog node. The fog node is responsible for collecting parking space information regarding occupancy and availability. It maintains a data structure, which is composed of a list of three-tuple: (parking Id, occupancy status, vehicleId).

1) parking Id (i): each parking spot is identified by a unique parking Id and is associated with the corresponding position (x_i, y_i) on the Euclidean plane.
2) occupancy status (OS): this indicates the status of the monitored parking space. It could be vacant or occupied, which can be represented in 2
   OS(i)= $\{$ 1 $if a vehicle is parked at parking spot \mathrm{p}_i$
   0 if the parking spot p_i is vacant
3) vehicle Id: this represents the vehicle's identification that occupies a parking space. This field is only used when the occupancy status value is 1; otherwise, this field will be empty.

To answer the request from the user whether there is a available parking space, we can simply find the value of $\exists i : OS(i) = 1$. This is only true if we do not accept reservations. In case the fog node allows drivers to reserve parking space ahead of time, we will need a more complicated data structure, which will be described in the next section, to answer the question of whether there is a parking space available.

The naming format of the interest message for requesting whether there is a available parking space can be: "application_type/Application_name/data name/timestamp/Geo-area." In this format, "Application_type" should be type A, indicating that it is an application interested in location-associated information. "Application_name" indicates it is a request for parking information, i.e., it is smart parking application; "Data name" represents the request is specifically requesting for parking lots with available parking spaces. "Timestamp" specifies the time the driver is interested in the availability information. Normally, it is the current time for requesting current availability. For the fog nodes that allow reservation, the query can also ask whether the parking lot has space available in some future time. This requires more complicated processing but can be accommodated with the reservation function discussed in the next section.

When a vehicle needs to look for a place to park, it can send a request as an interest packet for finding parking lots with available parking spaces. This request includes the geo-area information indicating the area the drive is interested in. This interest packet can be forwarded by other intermediate nodes using the algorithm proposed in the previous chapters to the desired target geographical area. The vehicles can use the V2V communication or take advantage of whatever infrastructure is available.

Upon receiving an interest message, an intermediate node will check its CS for matching content. If content with a matching timestamp is found, it will forward the corresponding data back to the vehicle making the request. Otherwise, it will check its PIT table for the same interest name. If the entry with the same interest name exists, the incoming interface of the interest will be added to the corresponding interfaces field in the PIT table. The interest entry in the PIT table will be removed if the entry timer expires.

When a fog node is in charge of a parking space in the target geo-area, it will process the interest packet and send back the data packet with related information. This data packet will be forwarded back to the vehicle making the request.

NDN fits perfectly with this function of the smart parking application because multiple drivers can send interest messages requesting the same content about available parking space. These requests will be aggregated along the way, except the fusion point will have multiple incoming interfaces, which will be used to forward data packets back to these drivers. All drivers can be satisfied with the same corresponding data packet from either the fog node or any caches at the intermediate nodes along the way.

D. Reserving a Parking Space

The other function of the smart parking application provide is to allow a driver to reserve a parking spot before his/her arrival from any location he/she currently is. The reservation will be based on the first comes first serve (FCFS) basis.

For the reservation application, we use the fog node to maintain the data related to parking spaces located at the same location and process the corresponding transactions locally. Also the fog node will use the central cloud as the backup service to maintain crucial states including existing user reservations. The central cloud also play a role of managing the setup of a new fog node and transition old states to it.

Multiple drivers may make reservations from the same parking lot with the same fog node. Each requester should be satisfied differently with the unique data content related to the request. NDN can satisfy multiple interest packets that carry the same name by fetching the original data from the producer or any intermediate node's corresponding data content. In this case, the data content will be the same for all the consumers. However, the case for reservation in the smart parking application is different in the sense that each requester should be satisfied differently, even though their interest packet may have the same name.

To deal with the issue, we need to make some modifications to NDN packets by extending the interest and data packet fields. The following fields are added to the interest packet: [my_id, my_arrival-time, my_departure-time, parkingSpace_location, my_position], where "my_ id" indicates the identification of the sender vehicle, "my_arrival-time" is the earliest possible arrival time, "my_departure-time" is the latest possible departure time, "parkingSpace_location" specifies the parking lot the driver wants to make the reservation, and "my_ position" indicates the sender vehicle position.

When an intermediate node receives the interest packet regarding the reservation, the interest name will be included in the PIT table if there are no matched data in the CS. The current NDN forwarding daemon has no concept about sending the same interest name and satisfying each consumer with different data packets. We introduce the concept of identity of vehicle interfaces (IVI) by modifying the PIT table to include the IVI, instead of the incoming interface field. The IVI entries include the binding of the incoming interface to the vehicle ID of the requester and a version number. This binding can be presented as <incoming interface, vehicleID, version>, where "version" indicates the version number when

a requester vehicle updates its route and sends a new version of the same interest. This binding helps to send the data packet to each requester vehicle separately. Once the intermediate node receives the data packet regarding a reservation request, the PIT will only satisfy the request with the same vehicleID. Only the IVI with the same vehicleID will be deleted. All other IVIs will remain there until the data packets matching their ID arrive. The PIT entry will be deleted if the data packet fails to return by the expiration time, or if the requests from all vehicles are satisfied.

When the reservation request arrives at the fog node responsible for a parking space, it extracts three tuple $< vid, r_a, r_d >$ from the request. vid is the id of the driver, r_a is the arrival time the driver expects to arrive to start parking the vehicle in the parking lot, and r_d is the expected departure time.

We assume that there are n parking spots in the parking space, denoted by $P = \{p_1, p_2, ..., p_n\}$, as shown in Fig. 2. The system will allow drivers to reserve for a certain period of time (e.g., a week, a month, etc.). The y-axis represents the n parking spots in the parking area. The x-axis indicates the time. Any reservation (as marked) is an interval on the x-axis.

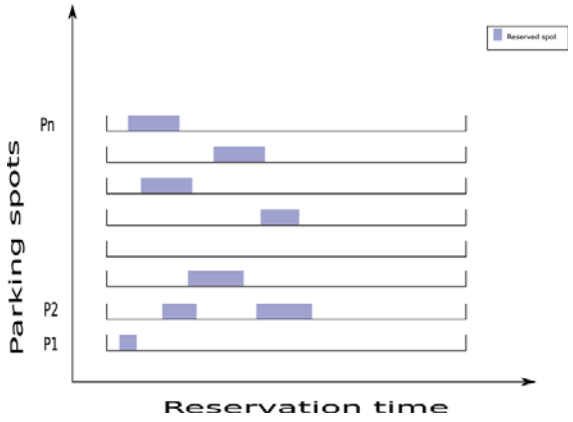

Fig. 2 Schedules for the Reservation System

The data structure represents the reservations, which is shown in Fig. 3. Any parking spot p_i consists of a list of reservations. Each reservation is represented by a node $< vid, a, d >$, where vid is the identification of the vehicle this spot is reserved for, a is the arrival time, and d is the departure time. $< a, d >$ represents an interval of the reservations. There is no overlap between all reservations for any given spot. To make the search algorithm for finding a spot that is available for the duration of a new request, we make sure that the list is sorted in increasing order based on the arrival time. If a spot has no reservation, the list is empty, and the pointer from that spot is NULL.

The system searches for a parking spot that is not reserved for the period of time $< r_a, r_d >$. If the system finds such an available interval, it reserves the parking spot for the requested duration, stores the reservation information and confirms the reservation. The reservation is in the format of $< vid, r_a, r_d, spot >$, where $spot$ is the parking spot reserved for a vehicle identified by vid. If no such spot is found, the system will report failure. Fig. 4 shows the detail of the

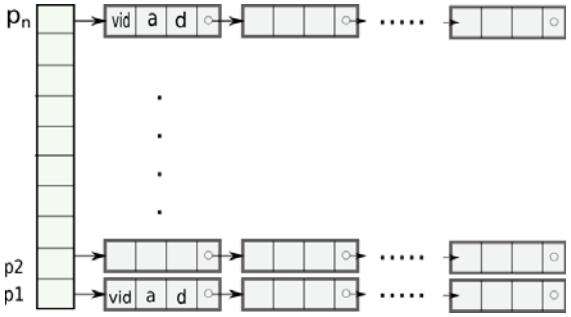

Fig. 3 Data structure for the reservation application

algorithm for reserving a parking spot. It goes through all the parking spots. If it finds a spot that has not been reserved by any vehicle, it will reserve the time interval (lines 2 to 5). Otherwise, it searches through the ordered list of current reservations. If the departure time is less than the arrival time of the first reservation for a node, the reservation will be added to the beginning of the list (lines 6-12). Otherwise, if it can find in the list two reservations such that the requested arrival time is after the first reservation's departure time and the requested departure time is before the second reservation's arrival time, the requested reservation will be put in between (lines 13 to 25). If we still cannot find a parking spot that is available for the duration of the requested reservation after going through all the parking spots, it returns failure.

Drivers can also cancel a previous reservation by sending the tuple $< vid, spot, r_a, r_d >$ to the fog node in charge of the parking space. Fig. 5 shows the algorithm for canceling a reservation. It first checks whether the parking spot number is valid or not (lines 1 to 3). If not, it will return with a failure. Otherwise, it searches the list of reservations for that spot (lines 4 to 18). If it finds a reservation that matches the user provided, it will delete the reservation from the list and return with success. If it cannot find the matching reservation, it will return with a failure (line 19).

E. Simulation Setup

For the simulations, we used ndnSIM (Version 2.8) [17], which is an ns-3 based NDN simulator. We also used SUMO [18], a microscopic traffic simulation for urban mobility, to generate the traffic. We created a map of a part of the University Campus. The map size is 2000×2000 meters. We generated different parking places distributed over the map. The vehicle speed limit is set to 20 km/h, and we set the transmission range of the signal of all vehicles to 150 meters. We utilized the ndnSIM parking applications as a starting point and implemented our smart parking application. The parameters for the experimental setup are shown in Table I. We assume that at each parking area, some parking spaces are occupied, and others are not.

F. Simulation Results

We evaluated the performance of an NDN-based smart parking application. Our evaluation consists of two parts. One

Data: the list of Parking Spots $P[1..n]$, the id of the vehicle making the request vid, request arrival time r_a, request departure time r_d
for $i = 1$ ***to*** n **do**
  if $P[i] == nil$ **then**
    $nptr$ = new $node(vid, r_a, r_d)$;
    $P[i] = nptr$;
    $return(success, i)$;
  else
    $cptr = P[i]$;
    if $r_d < cptr.a$ **then**
      $nptr$ = new $node(vid, r_a, r_d)$;
      $nptr.next = cptr$;
      $P[i] = nptr$;
      $return(success, i)$;
    else
      while $((cptr! = nil)$ *and* $(r_a >= cptr.d))$ **do**
        $qptr = cptr.next$;
        if $((qptr == nil)||(r_d <= qptr.a))$ **then**
          $nptr$ = new $node(vid, r_a, r_d)$;
          $nptr.next = qptr$;
          $cptr.next = nptr$;
          $return(success, i)$;
        else
          $cptr = qptr$;
        end
      end
    end
  end
end
$return(failure)$;

Fig. 4 Algorithm for Making a Reservation

TABLE I
SIMULATION STEP PARAMETERS

Maximum transmission range	150 m
Maximum transmission range of RSU	400 m
Number of vehicles	20, 30, ..., 100, 150
Vehicle speed	20 km/h
Simulation duration	200 ms
MAC type	IEEE802.11

is the evaluation of the function for finding parking lots with available parking spaces. The other is the evaluation of the parking reservation function of the system.

1) Finding the Availability of Parking Space: We compared our approach with the original NDN system. There are two main differences. One is how sensor nodes send the data to the fog node in charge of the parking space. In the original NDN, the sensor data will be sent to the fog node only if an interest message is broadcast from the fog node to the network. In our push-based model, after initial advertisement from the fog node, the sensor nodes can periodically push the sensed data to the fog node without the need to receive further interest packets.

Data: the list of Parking Spots $P[1..n]$, the id of the vehicle making the deletion request vid, the reservation $spot, r_a, r_d$, where $spot$ is the parking spot reserved, r_a is arrival time, r_d is the departure time
if $((spot < 1)||(spot > n))$ **then**
  return("Failure: Invalid spot number");
end
$cptr = P[spot]$;
$pptr = nil$;
while *(cptr)* **do**
  if $((cptr.v == vid)$ *and* $(cptr.a == r_a)$ *and* $(cptr.d == r_d))$ **then**
    if $(pptr == nil)$ **then**
      $P[spot] = cptr.next$;
    else
      $pptr.next = cptr.next$;
    end
    return("Success: Reservation deleted");
  else
    $pptr = cptr$;
    $cptr = cptr.next$;
  end
end
return("Failure: Reservation not found");

Fig. 5 Algorithm for Cancelling a Reservation

The other difference is how the vehicles make requests to find whether there are parking spaces available. In the original NDN, vehicles do not know any information about the fog node (an RSU or a selected parked vehicle). They will broadcast an interest message to request the availability of parking spaces. In our approach, vehicles are making the request to a specific geographical area and only those fog nodes in the geographical area may respond to the availability request.

We evaluate performance using two metrics, response time and push overhead.

1) Response time: defined as the time interval from the time when the vehicle makes the request by sending an interest packet to the time when it receives the response (data packet). The number of vehicles varies from 20 to 150.

Fig. 6 shows the response time when the number of vehicles varies from 20 to 150. We observed that the response time decreases when the number of vehicles increases. One reason is the caching effect of NDN because, with more vehicles, we can have a higher hit ratio at the cache at the intermediate nodes. The response time of our approach is less than that of the original NDN. Specifically, it achieves a 16.74% to 20.7% decrease when the number of vehicles increases from 20 to 150.

2) Push-based model overhead: defined as the total number of periodic *advertisement* messages that are broadcast by the fog node over the total number of pushed data packets to the fog node. Fig. 7 shows that the proposed push-based model achieves very low overhead, which decreases gradually as the number of vehicles increases. Establishing the connection

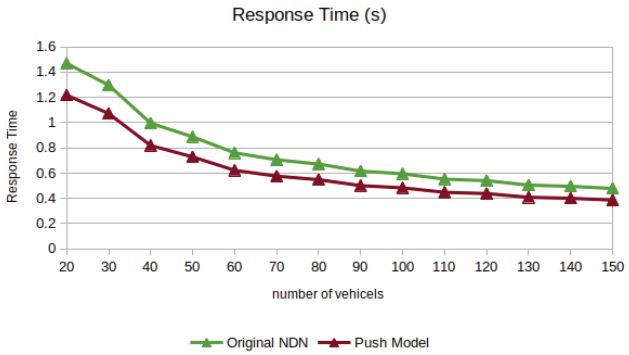

Fig. 6 Response time for different numbers of vehicles

by *advertisement* message helps in generating low overhead because we only need a few interest messages to be sent over the network. The push-based model outperforms the original NDN because we set a communication channel between the fog node and sensors to allow sensors to send periodically status information to the fog node.

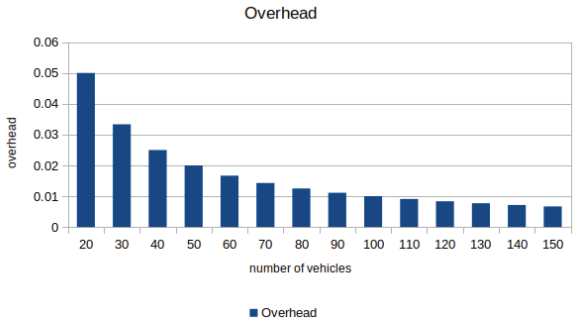

Fig. 7 Overhead caused by the proposed push-model

Reserving a parking space: We also evaluated the performance of the proposed reservation application by comparing it with a baseline approach, which allocates a parking space for a vehicle when it arrives at the parking lot without any prior reservation. The parameters for the experimental setup are shown in Table II. Fig. 8 illustrates the map generated by SUMO. Parking Areas are also identified on the map.

TABLE II
EXPERIMENT SETUP FOR PARKING SPACES

Parking slots	Total parking spaces
P1	25
P2	15
P3	30
P4	100
P5	60
P6	30

We evaluate performance using two metrics, success rate, and the average time to park.

1) Success rate to find a space: It is defined as the ratio of the number of vehicles that successfully find a vacant parking space over the total number of vehicles. Fig. 9 illustrates that when the number of vehicles increases from 20 to 150, the success rate to find a space decreases. Our reservation system has a higher success rate compared with the baseline approach. In the baseline approach, the driver just drives to a parking lot, and if the parking lot is full, the task of finding space is considered failed even if the driver may go to other parking lots and find space later. When the number of vehicles is low, the difference between the baseline approach and our reservation system is small. However, the difference becomes more obvious when the number of vehicles increases. We observed the success rates of our approach are 9.53%, 26.98%, and 53.08% higher than the baseline approach when the numbers of vehicles are 70, 110, and 150, respectively. Our approach can do better because it allows drivers to reserve a parking place before the arrival time.

2) Average time needed to park a vehicle: It is defined as the average time required to find a vacant parking space for a vehicle. Fig. 10 shows that the number of vehicles has an effect on the average time to park. As the number of vehicles increases, the average time needed to park a vehicle increases. From the figure, we can see that the average time to park for the reservation approach outperforms is smaller than that of the baseline approach. More specifically, they are between 40.36% and 53.31% less than those of the baseline approach.

IV. CONCLUSION

We categorized different types of V-NDN applications and provided an NDN naming scheme for each type. We considered a priority method to provide A better service for some time-sensitive applications. We developed an NDN-based framework for the smart parking application. A fog cloud architecture was adopted so that sensor data could be collected locally and requests could be handled through distributed fog nodes to reduce the load on a central cloud server. The cloud aspect allowed us to dynamically set up a new fog node from a static vehicle to provide service for those parking spaces without dedicated servers or RSUs. The smart parking application enables drivers to query about parking lots with available parking spaces and make a reservation.

Fig. 8 SUMO map for a smart parking application

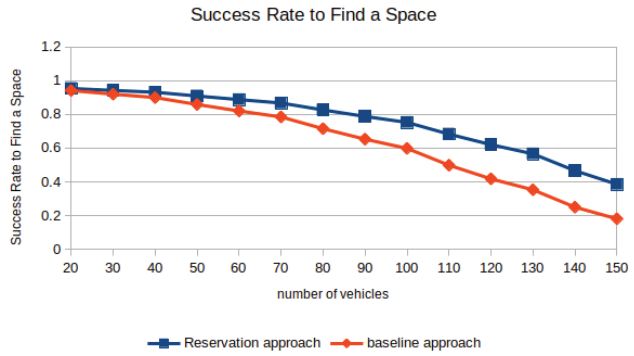

Fig. 9 Success rate of finding a space

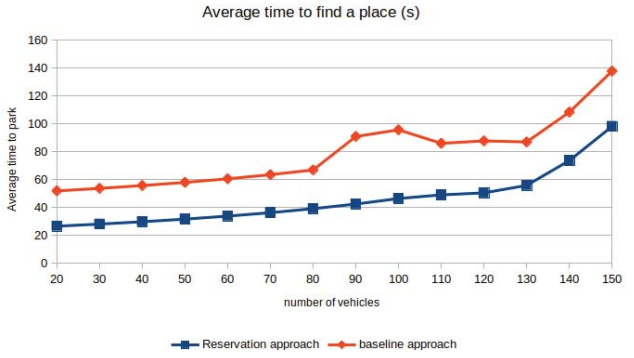

Fig. 10 Average time needed to park

REFERENCES

[1] Trista Lin, Herve Rivano, and Frederic Le Mouel. A survey of smart parking solutions. *IEEE transactions on intelligent transportation systems*, 18(12):3229–3253, 2017.

[2] X. Wang and S. Cai. An efficient named-data-networking-based iot cloud framework. *IEEE Internet of Things Journal*, 7(4):3453–3461, 2020.

[3] Khaoula Hassoune, Wafaa Dachry, Fouad Moutaouakkil, and Hicham Medromi. Smart parking systems: A survey. In *2016 11th International Conference on Intelligent Systems: Theories and Applications (SITA)*, 2016 11th International Conference on Intelligent Systems: Theories and Applications (SITA), pages 1–6. IEEE, 2016.

[4] IEEE Electrical Insulation Society Staff Corporate Author. Smart parking solutions for urban areas. In *2013 IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), pages 1–6, [Place of publication not identified], 2013. IEEE.

[5] Faheem, S.A. Mahmud, G.M. Khan, M. Rahman, and H. Zafar. A survey of intelligent car parking system. *Journal of Applied Research and Technology*, 11(5):714 – 726, 2013.

[6] Thierry Delot, Nicolas Cenerario, Sergio Ilarri, and Sylvain Lecomte. A cooperative reservation protocol for parking spaces in vehicular ad hoc networks. 01 2009.

[7] N Doulamis, E Protopapadakis, and L Lambrinos. Improving service quality for parking lot users using intelligent parking reservation policies. pages 1392–1397. IEEE, 2013.

[8] ElMouatezbillah Karbab, Djamel Djenouri, Sahar Boulkaboul, and Antoine Bagula. Car park management with networked wireless sensors and active rfid. In *Electro/Information Technology (EIT), 2015 IEEE International Conference on*, 2015 IEEE International Conference on Electro/Information Technology (EIT), pages 373–378. IEEE, 2015.

[9] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. *Fog Computing: A Taxonomy, Survey and Future Directions*, pages 103–130. Springer Singapore, Singapore, 2018.

[10] W. Balzano and F. Vitale. Dig-park: A smart parking availability searching method using v2v/v2i and dgp-class problem. In *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 698–703, 2017.

[11] Xueshi Hou, Yong Li, Min Chen, Di Wu, Depeng Jin, and Sheng Chen. Vehicular fog computing: A viewpoint of vehicles as the infrastructures. *IEEE transactions on vehicular technology*, 65(6):3860–3873, 2016.

[12] Y. Xiao and Chao Zhu. Vehicular fog computing: Vision and challenges. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 6–9, 2017.

[13] Chaogang Tang, Xianglin Wei, Chunsheng Zhu, Wei Chen, and Joel J. P. C Rodrigues. Towards smart parking based on fog computing. *IEEE access*, 6:70172–70185, 2018.

[14] Yi Zhang, Chih-Yu Wang, and Hung-Yu Wei. Parking reservation auction for parked vehicle assistance in vehicular fog computing. *IEEE transactions on vehicular technology*, 68(4):3126–3139, 2019.

[15] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, page 13–16, New York, NY, USA, 2012. Association for Computing Machinery.

[16] Ivan Stojmenovic, Sheng Wen, Xinyi Huang, and Hao Luan. An overview of fog computing and its security issues. *Concurrency and Computation: Practice and Experience*, 28(10):2991–3005, 2016.

[17] I. Moiseenko S. Mastorakis, A. Afanasyev and L. Zhang. ndnSIM 2: An updated NDN simulator for NS-3, ndn, technical report ndn-0028, revision 2, 2016, 2016.

[18] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.