THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique
École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par
Giulio GRASSI

Pour obtenir le grade de

DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Connected cars: a networking challenge and a computing resource for Smart cities

soutenue le 31 Octobre 2017

devant le jury composé de :

M. Giovanni Pau	Directeur de thèse
M. Mario Gerla	Rapporteur
M. Alexander Afanasyev	Rapporteur
M. Serge Fdida	Examinateur
M. Jérôme Härri	Examinateur
Mme. Giovanna Carofiglio	Examinatrice
M. Giovanni Pau	Examinateur
M. Mario Gerla	Examinateur
M. Alexander Afanasyev	Examinateur

Acknowledgments

After almost 4 years working at my PhD it is quite difficult to find the words to properly thank everybody. For sure I must thank my family for their support in all these years, before and during the PhD. My parents, my sister Erica and my little nephew Ian.

Of course I also have to thank Giovanni, my advisor, who has been not only a boss, but also a guide and a friend. We started working together in 2010, it seems ages ago. It won't be easy to get used to work with someone else.

Right after Giovanni comes Davide. We have been working together for four-five years, across different continents. It really feels weird knowing that I won't be working with you and Giovanni anymore. A lot of days and nights spent working, so many coffee breaks, a lot of nice memories, not only at work, but also outside the office.

Finishing with the Italian colony, thank to all the visitors who stopped by my office in these years, Andrea, Davide (or should I say Steve), and Edoardo. You didn't spend much time here, but it was fun hanging out with you guys.

Of course I can't forget the NPA team. All the PhD students, with some we started this adventure together, Benjamin, Alexandre, Quentin, others instead joined after of were there already, Matteo, Salah, Amr, Mustapha, Narcisse, Antonella, Florent. And of course the thanks don't stop to the students. Also all the professors, Serge, Marcelo, Prométhé, the list is quite long. Thank you everybody, it's was a great pleasure working here at NPA.

I can't forget the One-Lab crew, Pauline, Émilie, Radomir, Amira, Loic, Frederic, and Camille. You were there when the "Little Italy" in my office disappeared. A special thanks needs to be reserved to Émilie and Pauline, who helped me in fixing the French part of the thesis! In particular Pauline, who had to go through the entire introduction (twice).

During those years I've been collaborating with several people, each of them took some time from their busy schedule to teach me something. Lixia, Victor, Kyle, I learned a lot from you, thank you.

A PhD without paperwork and the help of the administration is not possible, so a big thanks to Sabine and Marguerite for all the assistance.

Finally, a big thanks to the rapporteurs, Mario Gerla and Alexander Afanasyev, who took their time to read my thesis and provided me with important feedbacks and suggestions, and to all the other members of the jury, Serge Fdida, Jérôme Härri and Giovanna Carofiglio.

After a long period it is never easy to thank everybody, there is always the risk of forgetting someone, especially in these frenetic moments, when 4 years of work and life have to be wrapped up. Hopefully I didn't forget too many person. It has been a great experience, not always easy, a PhD is never easy, but these 4 years have been great. I'm happy to finally be at the end of my PhD, but part of me is sad. I'll miss spending my days here at NPA.

Abstract

In the recent years we have seen a continuous integration of technology with the urban environment. This fusion aims to improve the efficiency and the quality of living in big urban agglomerates, while reducing the costs for their management. Cities are getting "smarter and smarter", with a plethora of IoT devices and sensors deployed all over the urban areas. Among those intelligent objects, an important role may be played by cars. Modern vehicles are (or will be) indeed equipped with multiple network interfaces, they have (or will have) computational capabilities and devices able to sense the environment.

However, smart and connected cars do not represent only an opportunity, but also a challenge. Computation capabilities are limited, mobility and the diversity of network interfaces are obstacles when providing connectivity to the Internet and to other vehicles. When addressing the networking aspect, we believe that a shift in the Internet model is needed, from a host oriented architecture (IP) to a more content focused paradigm, the Information Centric Networking (ICN) architectures. This thesis thus analyzes the benefits and the challenges of the ICN paradigm, in particular of Named Data Networking (NDN), in the VANET domain, presenting the first implementation running on real cars of NDN for VANET (V-NDN). It then proposes Navigo, an NDN based forwarding mechanism for content retrieval over V2V and V2I communications, with the goal of efficiently discovering and retrieving data while reducing the network overhead.

Networking mobility is not only a challenge for vehicles, but for any connected mobile device. For this reason, this thesis extends its initial area of interest — VANET — and addresses the network mobility problem for generic mobile nodes, proposing a NDN-based solution, dubbed MAP–Me. MAP-Me tackles the intra-AS content provider mobility problem without relying on any fixed node in the network. It exploits notifications messages at the time of a handover and the forwarding plane to maintain the data provider "always" reachable.

Finally, the "connected car" concept is not the only novel element in modern vehicles. Cars indeed won't be only connected, but also smart, able to locally process data produced by in-car sensors. Vehicles are the perfect candidates to play an important role in the recently proposed Fog Computing architecture. Such an architecture moves computational tasks typical of the cloud away from it and brings them to the edge, closer to where the data is produced. To prove that such a model, with the car as computing edge node, is already feasible with the current technology and not only a vision for the future, this thesis presents ParkMaster. Parkmaster is a fully deployed edge-based system that combines vision and machine learning techniques, the edge (driver's smartphone) and the cloud to sense the environment and tackle the parking availability problem.

Contents

List of Figures

List of Tables

Publications

G. Grassi, P. Bahl, K. Jamieson, G. Pau. ParkMaster: An in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments. *ACM-IEEE Symposium on Edge Computing (SEC)*. 2017

J.Auge, G. Carofiglio, G. Grassi, L. Muscariello, G. Pau, X. Zeng. MAP-Me: Managing Anchor-less Producer Mobility in Information-Centric Networks. *Under submission at IEEE Transactions on Network and Service Management; arXiv preprint arXiv:1611.06785*

J.Auge, G. Carofiglio, G. Grassi, L. Muscariello, G. Pau, X.Zeng. Anchor–less Producer Mobility in ICN. *ACM ICN Poster*. 2015

G. Grassi, M. Sammarco, P. Bahl, K. Jamieson, G. Pau. ParkMaster: Leveraging Edge Computing in Visual Analytics. *Mobicom Poster*. 2015

D. Pesavento, G. Grassi, G. Pau, P. Bahl, S. Fdida. Car-Fi: Opportunistic V2I by Exploiting Dual-Access Wi-Fi Networks. *Mobicom Demo*. 2015

G. Grassi, D Pesavento, G Pau, L Zhang, S Fdida. Navigo: Interest Forwarding by Geolocations in Vehicular Named Data Networking. WoWMoM 2015

G. Grassi, D Pesavento, G Pau, R Vuyyuru, R Wakikawa, L Zhang. VANET via Named Data Networking. IEEE INFOCOMM NOM Workshop. 2014

NFD developer's Guide. *Technical Report NDN-0021, NDN*

Introduction

The world population is moving to cities: as reported by the United Nations [170], while in the year 2014 54% of the population was urban, the estimations for 2050 say that two-thirds of the entire world's population will be living in cities. This constant growth and the complexity of city management is driving local governments towards the integration of technology with the city environment, to provide higher urban-life quality, better transportation and public services, while reducing the costs. For those reasons, the concept of smart city has gain popularity in the recent years. Frost and Sullivan [49] forecast that by the year 2025 more than thirty cities around the world will become smart cities or will have implemented some of the smart city's features (Figure 1.2). While there is no universally accepted definition for "smart city" and plenty of variations are present in the literature [12], a good definition is provided by [26], which defines "smart cities" as a technological intensive and advanced city that connects everything and everyone, people, information, urban objects etc. in order to create a more efficient, green and sustainable urban environment and thus providing a higher quality of city-life (Figure 1.1 shows some examples). A key role in such integration between city's infrastructure and intelligent devices is played by the Internet of Things. An urban IoT deployment is indeed able to help monitoring and controlling the status and the efficiency of public areas, parking, public transport systems, traffic, garbage collection etc., making the city not only smart, but also connected.

One of the aspects that the design of smarter cities takes into consideration is the private and public transportation, from increasing the travel efficiency to decreasing accidents and costs and making the transportation experience more conformable. At the same time, cars industry is investing large amount of money on the concept of smart and connected car, designing vehicles with networking and computation capabilities. [159] forecasts that the connected and smart car market (including the related services) will be worth more than $150 billion by 2022, while BI Intelligence estimates that by the year 2020 the yearly production of connected cars will be over 50 million (Figure 1.3). These two aspects together

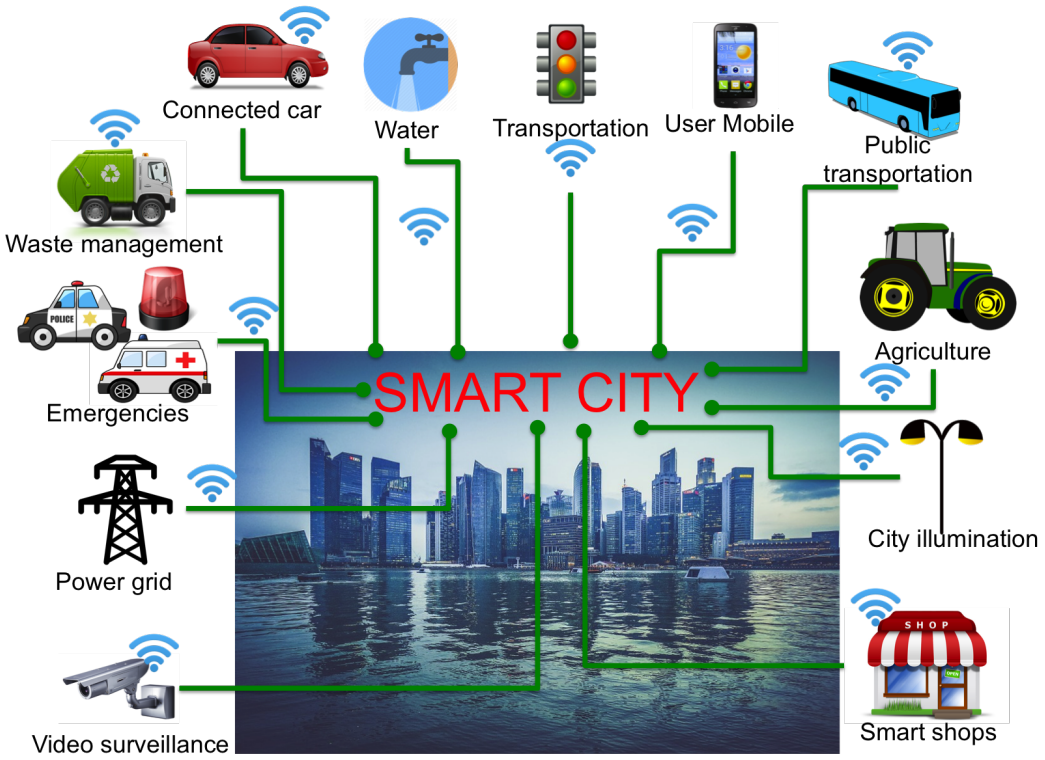


FIGURE 1.1: Smart city: Use cases

lead to a clear and natural conclusion: vehicles will play a key role in the smart city of the future.

Cars will become at the same time information provider, service consumer, sensors and actuators. This integration will lead to the next technological transformation that will enable a wide range of new personal services in which cars will play different roles: vehicles will be able to collect a large and valuable quantity of information about the world that surrounds them, consume data locally produced or available on the Internet, and finally, they will be the way this information will be propagated, sharing data they collect but also helping propagating the information other vehicles are collecting, being the carrier and the forwarder of the data. Such scenario enables new use cases and new possibilities for applications, but it also brings new challenges to the table, such as high node mobility, communication disruptions etc.

Finally, an additional role may be played by cars. With the advancements in computation and the reduction of its costs, a new concept has gained popularity in the past few years: **Fog computing**. Instead of always relying on far away nodes like the cloud, this concept places applications, data and processing at the logical extremes of a network. By placing data and data-intensive applications at the edge instead of some central node, the amount of data that needs to be

FIGURE 1.2: Smart cities in the world.

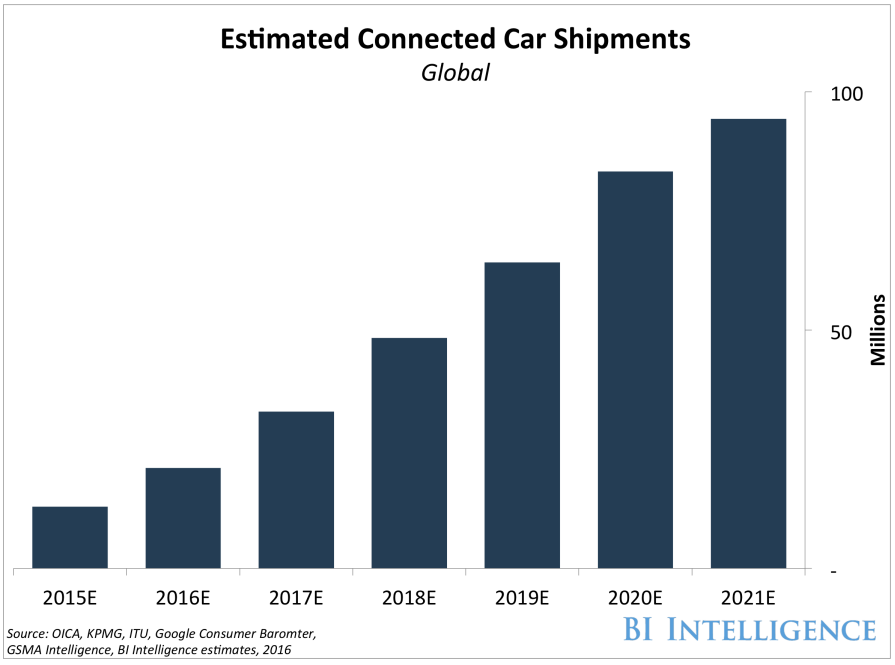


FIGURE 1.3: Connected cars shipments forecast.

moved across the network and the traveled distance are reduced. Thus, with cars already in posses of networking and computation capabilities, they are a natural candidate to become edge nodes and process data in loco, without the constant need of the cloud, spreading computation capabilities all over the city.

Summarizing, in the smart city of the future cars will be able to produce data and sample the surrounding environment, consume local and global data, help

others reaching their content and locally process large amount of data.

Challenges Even though smart cities are not far in the future and the connected and intelligent car is almost a reality, there are still several challenges to be addressed before we can see a fully operable smart city with vehicles playing all the aforementioned roles.

First of all, the "connected car" concept presents several challenges, due to a highly mobile environment and a heterogeneity of wireless technologies. Recently manufactured vehicles are indeed equipped with a variety of wireless communication interfaces such as 3G/LTE, WiMAX, WiFi, IEEE1901 (Power Line Communication), and 802.11p (DSRC/WAVE). Ideally a car should be able to utilize any and all of these interfaces to communicate with either infrastructure servers or other vehicles as needed by applications. Whenever more than one interface is available, the vehicle should be able to pick and choose the best one or use multiple in parallel. However, the increasing number of vehicles that are connected to Internet today, are mainly connected via cellular networks *only*. The concept of connecting vehicles through Road-Side-Units (RSUs) has long existed, but only certain regions or countries deployed them. Standards have been developed for direct Vehicle-to-Vehicle (V2V) communication, however the usage is limited to one-hop communication for collision prevention only. There have been numerous publications exploring the use of V2V to support a much broader range of applications, unfortunately those results are isolated point solutions using various patches to overcome the limitations of TCP/IP, hence lacking a basic framework to prove their utility. We believe indeed that the root cause of this insoluble problem in vehicles networking that all those solutions face resides in IP's communication model. In IP's paradigm indeed, the endpoints, with their addresses, are the center of the communication. Data packets are encapsulated into IP datagrams carrying the addresses of the endpoints, without any connection to the data itself, which is hidden inside the datagram's payload. Even though what is relevant to the users is the data exchanged, the IP's model ignores this and focuses on the endpoints of the communication itself. Due to this host centric characteristic of IP, the communication becomes susceptible to any change of the endpoint's network interface. A car moving and changing the WiFI access point is associated with, or a momentarily switch from WiFi to 3G, breaks (or at least deteriorate) the connection.

While this seems an insoluble problems with IP-based approaches (without any patch), a new network paradigm may bring the solution within its design: the Information Centric Networking paradigm (ICN). This recent network architecture family moves from the host-centric paradigm of IP to a

"information-based" paradigm, where each piece of information users want to exchange is named. The focal point of the communication model is not the endpoint nor the connection among two hosts, but is the exchanged content itself and its name. The latter identifies the information exchanged and detached it from the bit exchanged: data survives in the network after it has been transferred and can be retrieved thanks to its name — the network is not anymore a pipe that transfers anonymous series of bits, but a system of forwarding nodes able to cache data and make it available afterwards by name, transparently, as if it is produced by the node. In this architecture data becomes an independent object. It does not depend on locations, applications and means of transportation. The connectivity may be intermittent, mobility and multi access are the norm, anycast, multicast and broadcast are natively supported. Finally, security is embedded within the data itself. Data integrity and authenticity don't depend on the channel used, but can be verified by any node in the network given the data packet and some public information (i.e. public key) of the producer.

Those factors lead us to believe that a shift in the Internet paradigm from the IP's host-centric model to the ICN information-centric model can be the definitive solution for the diffusion of vehicular network in smart cities (and in general can bring great benefits to the rest of Internet). Thus, we apply the Named Data Networking paradigm, one of the ICN family architecture, to the vehicular network, demonstrate its feasibility and efficacy on the field, evaluate the challenges and address some of the unresolved questions, like how to efficiently enable data retrieval through multi-hop V2V communication and how to exploit NDN characteristics like caching and data naming to reach the desired data.

The information-centric's benefits however don't stop at vehicular networks. The natural support for mobility offered by ICN indeed makes it a good candidate to define a radically new solution for the more generic mobility-management problem and to provide a native integration into 5G networks, overcoming limitations of the traditional IP-based approaches. If consumer mobility is supported in ICN by design, in virtue of its connectionless pull-based communication model, producer mobility is still an open challenge. We thus extended our work to the more general network mobility problem and designed a NDN based solution to maintain data reachability when the node producing the content is a mobile node (thus not only car, but also other mobile devices as smartphone, tablet etc.).

Computing at the edge: As previously mentioned, networking is not the only aspect involved in the role of vehicles in the smart city of the future. Indeed,

connectivity, together with computing capabilities, enable vehicles to not only produce or consume data, but also to process it locally, becoming at all effects mobile computing nodes, acting as edge nodes in a Fog architecture. This recent paradigm proposes to exploit computing capabilities of nodes closer to user, closer to where data is produced and consumed — the edge of the network — to process data, instead of always relying on the cloud. The goals of such a paradigm are reducing the connectivity/bandwidth requirements for the nodes — less data needs to be transfered to the cloud and permanent connectivity to the Internet may not be required — scalability, smaller delays (data transfer delay is zero). Services, applications and computing tasks are pushed from far away centralized point — the cloud — to nodes closer to the user — the edge. As shown by Figure 1.4, typical examples where the edge computing approach can be successful might be video surveillance data stream processing (e.g. vision algorithms running at the edge to analyze video streams and upload to the cloud only small summary to the cloud); user's device support (e.g. smart-glasses or smartphones may produce large quantity of video-data for augmented reality tasks that needs to be processed in timely manner); traffic control (e.g. edges close to the intersection collect real-time information about the traffic status and promptly react to handle traffic more efficiently or to avoid accidents); etc.

Within the transportation domain, cars can play an important role in the edge computing architecture. In-car devices and sensors and driver's devices (i.e. smartphone) can indeed produce a huge amount of data while sensing the environment (i.e. information about location, telemetry, images and videos etc.) and, even if the car is well connected, it's unfeasible (or too expensive) to completely rely on the cloud and upload everything. With in-loco computing capabilities, thanks to car's on-board smart unit (OBU) or user's devices, a new opportunity presents: local data can be processed and consumed locally, and then, when needed, a (optionally smaller) version of it can be shared with other vehicles or smart objects in the area with direct communication (i.e. V2V) or uploaded to the cloud to make the information available to everyone.

With the progress in the computation capabilities of small mobile devices like smartphones, the edge running within a car is not only a vision for the smart city of the future, but a reality for the city of today. We believe indeed that, while in the future in-cars devices will keep evolving and getting more efficient, we already have all the components to start exploiting such a model — the current technology is ready to bring computation capabilities close to the city and to its habitants. For instance, we believe that with the plethora of different sensors available inside a car, is already possible to sense the environment and capture valuable information in loco — at the edge — without the need of

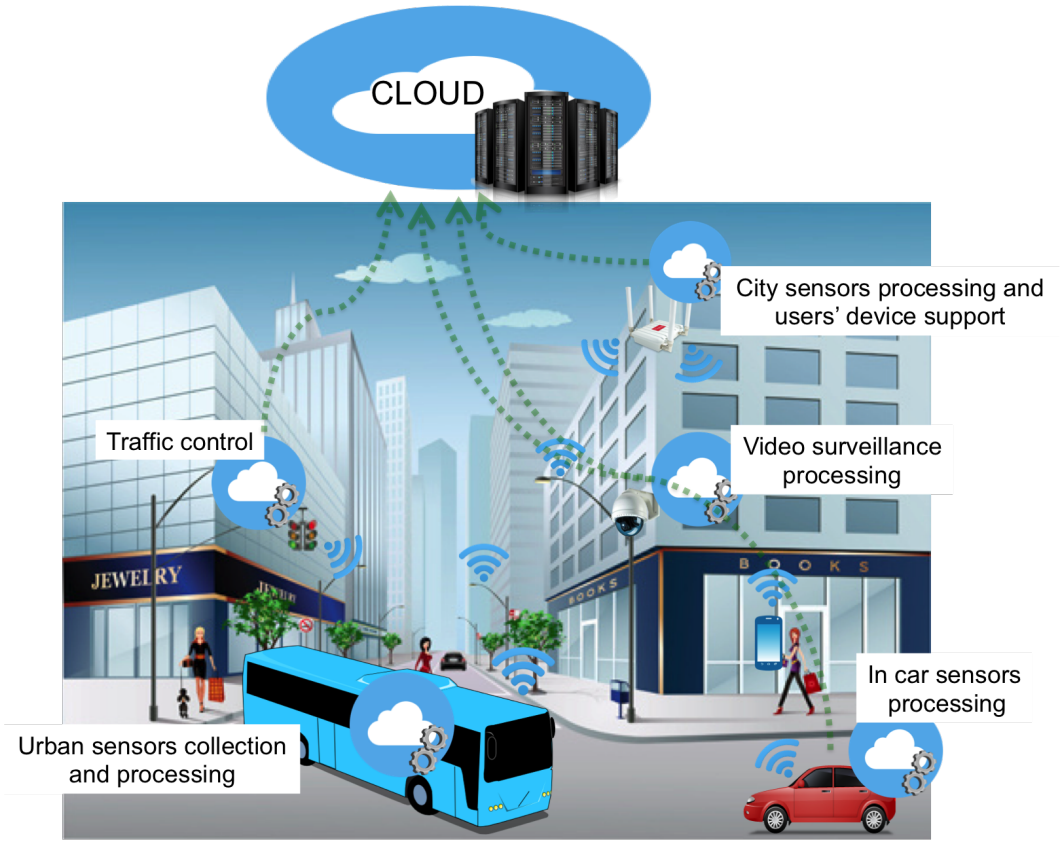


FIGURE 1.4: Fog computing in smart city.

remote "super computers", but simply relying on local devices' computing capabilities, like, for instance, user's devices as smartphones. Being able to sense the environments can indeed enable an enormous number of applications, from safety (i.e. avoid cars or pedestrian accidents), real-time information about traffic, roads status, parking availability etc. Especially considering the advancements in vision algorithms and vision processing, one would expect to be able to exploit the ubiquitous cameras (vehicles, video surveillance systems, smartphones etc.) to make the next step in the urban environment sensing and integrate traditional sensors systems with more rich vision-based approach. This however, with cloud-based solutions, in particular when dealing with mobile nodes, encounters several limitations because it requires continuous connectivity to the Internet and an enormous bandwidth to upload data to the cloud to process it. In contrast, in the fog-based architecture we envision, current local devices' computing capabilities (as the user's smartphone has) allow to remove those obstacles and make video-based on-car urban sensing available with the current technology, as work presented in this thesis demonstrates.

Contributions This thesis collects personal and collaborative work done during my PhD program (the full list of co-authors is available in the "Publications" section), resulting in the following contributions:

- The design and implementation of the first "NDN forwarder for vehicular networks" running on real hardware (Chapter 3).
- The proof by on-the-field experimentations of the feasibility and the benefits of NDN for VANET and the identification of the remaining challenges (Chapter 3).
- The design of a mapping system between data names and data location to facilitate the utilization of geographic forwarding techniques to the data retrieval problem in VANET (Chapter 4).
- A road-topology based geographic forwarding mechanism for V2V (over NDN) (Chapter 4).
- The design of an adaptive discovery and selection mechanism able to identify the available data sources across multiple geographic areas and quickly react to sudden changes in vehicle networks (Chapter 4).
- The proposal of a mobility-management mechanism that addresses the micro (e.g. intra Autonomous Systems) producer mobility problem in NDN, completely distributed and without relying on global routing updates or on the assistance of some external network components (Chapter 5).
- The running proof that the current technology is ready for running image-based machine learning techniques in the in-car edge, on today's smartphones, to capture valuable information about the surrounding environment in real-time: the design, implementation and deployment of a complete and running edge-based system that tackles the available on-the-road parking location problem (Chapter 6).
- A novel lightweight tracking algorithm for car detection that uses location estimates to "de-duplicate" multiple detections of the same parked vehicle in a drive (Chapter 6).
- A new localization algorithm to estimate parked car location with a single frame, without users interventions nor stereo vision (Chapter 6).

Thesis Outline In Chapter 2 a brief overview of Named Data Networking is presented, followed by the state of the art description of Vehicular networks (in particular routing and forwarding mechanism), node mobility solutions and Fog

Computing. Later, Chapter 3 presents the description of the first attempt of running NDN for VANET (dubbed V-NDN), while the evolution of its forwarding scheme is presented in Chapter 4. Afterwards, MapMe, an NDN-based approach to the producer mobility problem (Chapter 5) is presented. Finally, Chapter 6 describes the design and evaluation of ParkMaster, a visual-analytics based edge-system for the available parking spot search problem.

Introduction en Français

La population mondiale se déplace vers les villes: comme rapporté par les Nations Unies [170], alors qu'en 2014, 54% de la population était urbaine, les estimations pour 2050 indiquent que les deux tiers de la population mondiale vivront dans les villes. Cette croissance constante et la complexité de la gestion de la ville poussent les gouvernements locaux à l'intégration de la technologie avec l'environnement de la ville, dans le but de fournir une meilleure qualité de vie urbaine, de meilleurs transports et services publics, tout en réduisant leurs coûts. Pour toutes ces raisons, le concept de ville intelligente (smart city) a gagné en popularité ces dernières années. Les prévisions pour l'année 2025 indiquent que plus de trente villes dans le monde atteindront alors leur transformation en villes intelligentes ou auront mis en place certaines fonctionnalités de la ville intelligente (Figure 1.6). Bien qu'il n'y ait pas de définition officielle de la ville intelligente et que de nombreuses variantes soient présentes dans la littérature [12], une définition est donnée par [26], qui définit la "smart citiy" comme une ville à la technologie avancée connectant l'ensemble des gens, des informations et des objets urbains dans le but de créer un environnement urbain plus efficace, écologique et durable et de fournir ainsi une meilleure qualité de vie citadine (quelques exemples Figure 1.5).

L'Internet of Things joue un rôle clé dans l'intégration de dispositifs intelligents au sein de l'infrastructure d'une ville. Un déploiement urbain de l'IoT a la capacité d'aider à surveiller et à contrôler l'état et l'efficacité des espaces publics, du stationnement, des systèmes de transport public, de la circulation, de la collecte des ordures, etc. Des aménagements qui rendent la ville non seulement intelligente, mais aussi connectée.

La conception de villes plus intelligentes passe par le transport, qu'il soit privé ou public: depuis une plus grande efficacité du voyage à la diminution des accidents et des coûts en passant par une expérience de transport plus agréable, celui-ci est un paramètre essentiel. Dans le même temps, l'industrie automobile

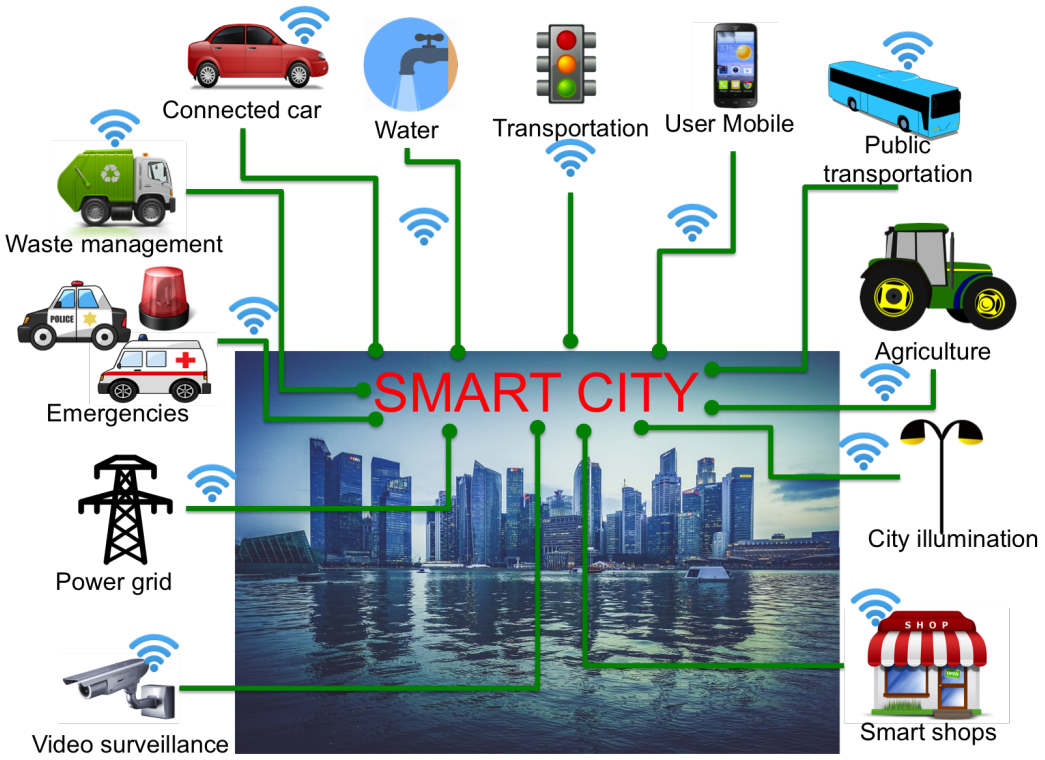


FIGURE 1.5: Smart city: cas d'utilisation

investit massivement dans le concept de voiture intelligente et connectée, en concevant des véhicules aux capacités de réseautage et de calcul importants. En 2022 [159] prévoit une valorisation du marché des voitures intelligentes et connectées à 150 milliards de dollars, alors que en 2020 il y aura une production annuelle de plus de 50 millions voitures selon BI Intelligence (Figure 1.7) Ces chiffres nous amènent à une conclusion évidente: les véhicules joueront un rôle clé dans la ville intelligente du futur.

Les voitures seront à la fois fournisseur et consommateur d'information, capteurs et actionneurs. La prochaine transformation technologique, annoncée par cette intégration des voitures dans la ville, permettra le lancement d'une large gamme de nouveaux services dans lesquels les automobiles joueront différents rôles. Les véhicules pourront collecter une quantité importante d'informations sur le monde qui les entoure et consommer ces données. Qu'elles soient produites localement ou disponibles sur Internet, ces données seront donc propagées par les véhicules eux-mêmes qui pourront les collecter depuis un autre véhicule, les transporter, les partager. Un tel scénario ouvre de nouveaux champs d'utilisation et de nouvelles possibilités pour les applications. Mais ces bouleversements nous amènent également à faire face à de nouveaux défis, comme l'intense mobilité des nœuds ou les perturbations de la communication.

FIGURE 1.6: Smart cities dans le monde.

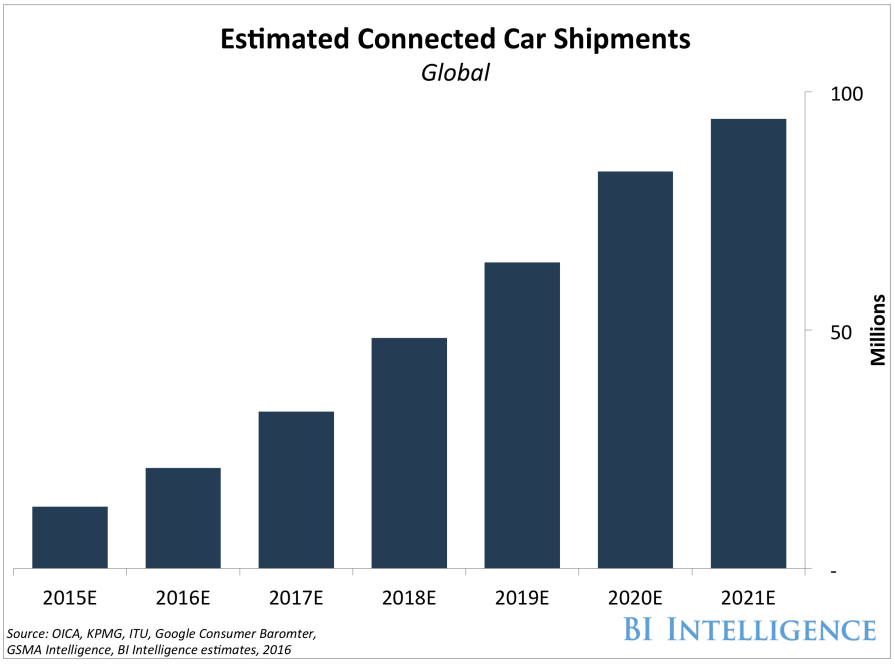


FIGURE 1.7: Prévisions des ventes de voitures connectées.

Enfin, un dernier rôle peut être joué par les voitures. Avec les progrès effectués dans les calculs et la réduction de leurs coûts, un nouveau concept a gagné en popularité au cours des dernières années: le Fog-computing. Les applications, les données et leur traitement sont placés aux extrêmes logiques d'un réseau

contrairement à une utilisation du cloud qui se base sur des nœuds distants. En plaçant des données et des applications à forte intensité de données dans l'edge et non dans un nœud central, la quantité de données qui doit être déplacée sur le réseau ainsi que la distance parcourue sont réduites. Les voitures connectées et possédant de fortes capacités de calcul, sont des candidats idéals pour devenir des edge-node. Elles peuvent traiter des données en local, sans un besoin constant de cloud, et diffuser leurs capacités de calcul dans toute la ville.

En résumé, dans la ville intelligente du futur, les voitures serviront à capturer des échantillons d'information provenant de leur environnement, produire et consommer des données locales et distantes, à aider d'autres véhicules à recevoir le contenu souhaité et à traiter localement une grande quantité de données.

Défis Bien que les villes intelligentes soient aux portes de demain et que la voiture connectée soit déjà bientôt une réalité, il reste plusieurs défis à relever avant qu'une ville intelligente soit entièrement opérationnelle et dans laquelle les véhicules joueraient un rôle prépondérant.

En raison d'un environnement hautement mobile et d'une hétérogénéité des technologies sans fil, le concept de voiture connectée présente plusieurs défis. Les véhicules récents sont en effet équipés d'une variété d'interfaces de communication sans fil telles que 3G/LTE, WiMAX, WiFi, IEEE1901 (Power Line Communication) et 802.11p (DSRC / WAVE). Selon les besoins des applications, une voiture devrait pouvoir utiliser toutes ces interfaces pour communiquer avec des nœuds sur Internet ou avec d'autres véhicules. Lorsque plus d'une interface est disponible, le véhicule devrait pouvoir choisir le meilleur ou en utiliser plusieurs en parallèle. Cependant, le nombre croissant de véhicules qui sont actuellement connectés à Internet sont la plupart du temps uniquement connectés via des réseaux cellulaires. Le concept de connexion des véhicules par les unités de route (RSU) existe depuis longtemps, mais seules certaines régions ou pays les ont déployés. Les standards ont été développées pour la communication directe véhicule-véhicule (V2V), mais l'utilisation est limitée à un "single-hop communication" pour parer à d'éventuelles collisions. De nombreuses publications ont exploré l'utilisation de V2V pour supporter une gamme beaucoup plus large d'applications. Malheureusement les résultats n'ont apporté que des solutions isolées qui utilisent différents correctifs pour surmonter les limitations de TCP/IP, sans solution générique. Nous croyons en effet que la cause de ce probléme soit le modèle de communication IP: dans le paradigme de l'IP, les points d'extrémité et leurs adresses, sont le centre de la communication. Les paquets de données sont encapsulés dans les datagrammes IP portant les adresses des points d'extrémité, sans aucune connexion aux données elles-mêmes,

cachées à l'intérieur du datagramme. L'IP ne tient pas compte de la donnée échangée, même si celle-ci est essentielle pour l'utilisateur, et se concentre sur les points d'extrémités de la communication elle-même. A cause de cette caractéristique centrée sur l'hôte, la communication est sensible à toute modification de l'interface de réseau du point d'extrémité. En se déplaçant, une voiture qui change le point d'accès WiFi auquel elle est associée ou passe momentan du WiFi à la 3G casse, ou du moins détériore, la connexion.

Alors que les approches basées sur l'IP (sans aucun hack) semblent difficilement exploitables, un nouveau paradigme de réseau apporte une solution dans son design: le paradigme Centric Networking Information (ICN). Contrairement au concept centré sur l'hôte (IP), cette récente famille de protocole de réseau est basée sur l'information. Cette dernière, que l'utilisateur souhaite échanger, est nommée. L'élément principal de ce modèle de communication n'est pas le point d'extrémité ni la connexion entre deux hôtes, mais le contenu échangé lui-même et son nom. Celui-ci identifie les informations échangées et les détache du bit échangé: les données persistent dans le réseau après leur transfert et peuvent être récupérées grâce à leur nom — le réseau n'est plus un tuyau qui transfère des séries de bits anonymes, mais un système de relais de nœuds capables de stocker temporairement en cache des données et de les rendre disponibles par la suite, de manière transparente, comme si elles étaient produites par le nœud. Dans cette architecture, les données deviennent un objet indépendant qui ne dépend plus de l'emplacement, des applications ou des moyens de transport. La connectivité peut être intermittente, la mobilité et les accès multiples sont la norme. Enfin, anycast, multicast et broadcast sont supportées de façon native.

Ces facteurs nous amènent à croire que le passage d'un modèle IP centré sur l'hôte à un modèle ICN centré sur l'information semble être la meilleure solution à la diffusion d'un réseau de véhicules dans les villes intelligentes, et peut, de façon plus générale, apporter d'excellents avantages au reste d'Internet. Ainsi, en appliquant le paradigme Named Data Networking, l'un des protocoles de la famille ICN, au réseau de véhicules, nous avons démontré sa faisabilité et son efficacité sur le terrain, évalué les défis et abordé certaines des questions non résolues, comme la façon d'activer efficacement la récupération de données avec la communication multi-hop V2V et la façon d'exploiter les caractéristiques types du NDN comme le caching et le nom de la donnée pour obtenir les informations souhaitées.

Cependant, les avantages de l'ICN ne s'arrêtent pas au réseau de véhicules. L'ICN supporte nativement la mobilité des nœuds, ce qui en fait un candidat idéal pour définir une solution radicalement nouvelle pour le problème de gestion de la mobilité. Il permet également de fournir une intégration native dans les

réseaux 5G, en surmontant les limites des approches traditionnelles basées sur l'IP. Si la conception même du protocole ICN supporte la mobilité des consommateurs, en raison de son modèle de communication sans connexion, la mobilité des producteurs reste un défi à relever. Nous avons donc élargi nos recherches au problème plus général de la mobilité du réseau et conçu une solution basée sur NDN pour maintenir l'accessibilité des données lorsque le nœud produisant le contenu est un nœud mobile. Et pas seulement une voiture, mais aussi d'autres appareils mobiles tels qu'un smartphone ou une tablette.

Calcul dans l'edge: Comme mentionné précédemment, le réseautage n'est pas le seul aspect impliqué dans le rôle des véhicules dans la ville intelligente du futur. En effet, la connectivité et la capacité de calcul permet aux véhicules non seulement de produire ou de consommer des données, mais aussi de les traiter localement. Ils deviennent ainsi des nœuds de calcul mobiles, agissant comme des edge-node dans une architecture de Fog-computing. Contrairement à un modèle basé sur le cloud, ce récent paradigme propose d'exploiter les capacités de calcul des nœuds les plus proches à la fois de l'utilisateur, de l'origine des informations et d'où elles sont consommées — le edge du réseau — pour traiter les données. Les objectifs d'un tel paradigme réduisent les exigences de connectivité/bandwidth pour les nœuds: une quantité moindre de données doit être transférée sur le cloud et une connectivité continue à Internet n'est plus un prérequis, ce qui induit une scalabilité et un délais inférieur (le délai de transfert de données est null). Les services, les applications et les tâches du calcul sont poussés du point centralisé distants — le cloud — aux nœuds les plus proches de l'utilisateur — l'edge. Qu'il s'agisse du traitement de flux de données de vidéosurveillance, avec l'exécution des algorithmes de computer-vision dans l'edge pour analyser les flux vidéo et télécharger uniquement sur le cloud un résumé des données; du support de l'appareil de l'utilisateur, avec la grande quantité de données vidéo générée par les smart-glasses ou les smartphones pour les tâches de réalité augmentée et qui doivent être traitées en temps réel; du contrôle de la circulation, avec un edge proche de l'intersection qui recueille des informations en temps réel sur l'état de la circulation et doit réagir rapidement pour traiter le trafic plus efficacement ou pour éviter des accidents, ces différents exemples démontrent le succès de l'approche du edge-computing (Figure 1.8).

Dans le domaine des transports, les voitures peuvent jouer un rôle prépondérant dans l'edge computing. Les périphériques et les capteurs dans la voiture et les périphériques du conducteur, comme son smartphone, peuvent effectivement produire une grande quantité de données tout en détectant l'environnement (informations sur l'emplacement, télémétrie, images, vidéos etc.). Et même si la

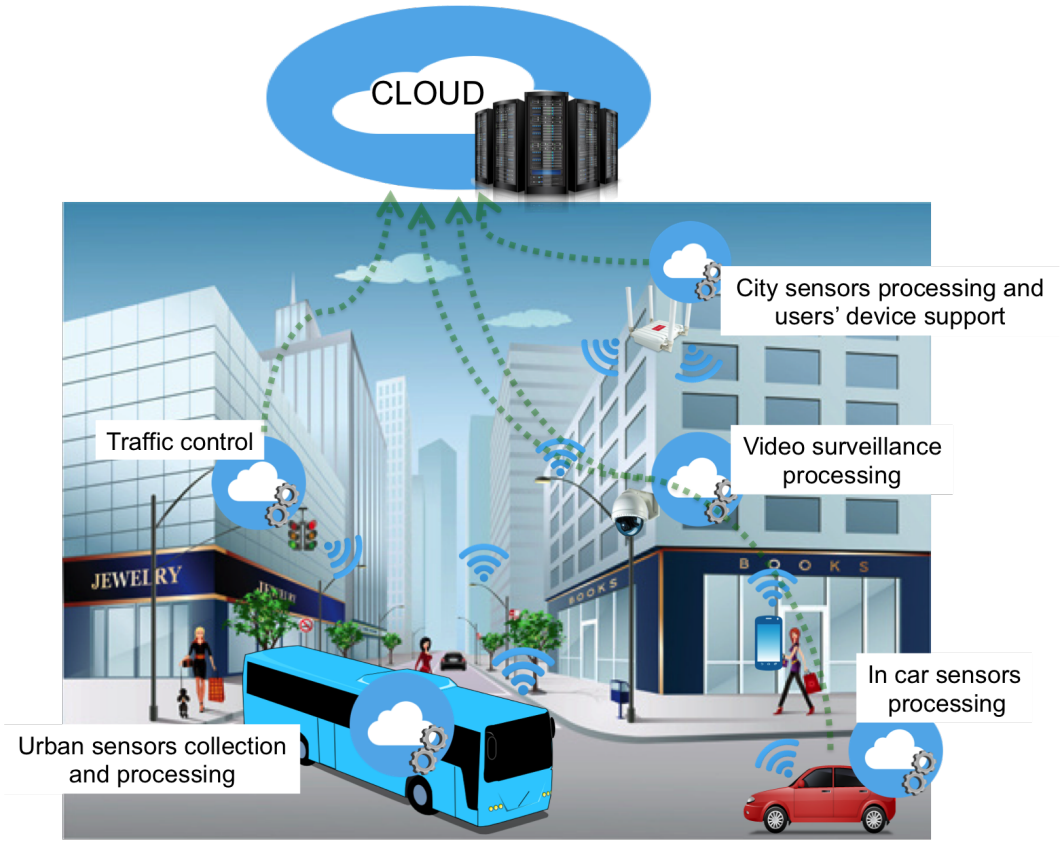


FIGURE 1.8: Fog computing dans le smart city.

voiture est correctement connectée, il est impossible, ou trop coûteux, de dépendre entièrement du cloud et de télécharger l'intégralité des données. Avec les capacités de calcul in loco obtenues grâce à l'unité intelligente embarquée de la voiture (OBU) ou aux périphériques de l'utilisateur, une nouvelle opportunité se présente: les données locales peuvent être traitées et consommées localement, puis, le cas échéant, une version, éventuellement plus légère, peut être partagée avec d'autres véhicules ou avec des objets intelligents proches grâce à une communication directe (i.e. V2V) ou téléchargés sur le cloud pour rendre l'information à la portée de tous.

Avec le progrès des capacités de calcul des petits appareils mobiles comme les smartphones, faire de l'edge-computing dans une voiture n'est pas seulement une vision pour la ville intelligente du futur, mais une réalité pour la ville d'aujourd'hui. En effet, alors que dans le futur les appareils automobiles continueront d'évoluer et de devenir toujours plus efficaces, nous avons aujourd'hui tous les composants pour commencer à exploiter un tel modèle: la technologie actuelle est prête à apporter des capacités de calcul proches de la ville et de ses habitants. Par exemple, avec la quantité de différents capteurs disponibles à l'intérieur d'une voiture, il est à présent possible de détecter

l'environnement et de capturer des informations précieuses in loco — sur l'edge — sans avoir besoin d'un "super ordinateur" à distance, mais simplement en s'appuyant sur les capacités de calcul des périphériques locaux, tel que le smartphone d'un utilisateur. Avec l'analyse des environnements urbains, un grand nombre d'applications est possible dans tous les domaines : la sécurité (i.e. éviter les accidents de voitures et de piétons), les informations en temps réel sur la circulation, l'état des routes, la disponibilité du stationnement, etc. Et L'edge est essentiel pour permettre le développement de ces services.

Concernant les progrès des algorithmes de computer-vision, on s'attend principalement à exploiter l'omniprésence des caméras (véhicules, systèmes de vidéosurveillance, smartphones etc.) et ainsi passer à la prochaine étape de l'analyse de l'environnement urbain: l'intégration des systèmes de capteurs traditionnels vers des systèmes basées sur le computer-vision. Les solutions basées sur le cloud, en particulier lorsqu'il s'agit de nœuds mobiles, sont soumises à plusieurs contraintes dû à la nécessité d'une connectivité continue à Internet et d'une immense bandwidth pour télécharger des données sur le cloud afin de les traiter. Dans une architecture basée sur le fog-computing et comme cette thèse tentera de le démontrer, la technologie et les capacités de calculs actuelles des périphériques locaux permettent de supprimer ces obstacles et de réaliser une détection sensorielle urbaine basée sur le video-processing dans les voitures.

Contributions Les contributions de cette thèse sont les suivantes :

- La conception et la mise en œuvre du premier "NDN forwarder pour les réseaux de véhicules" fonctionnant sur des réel voitures (Chapitre 3).
- La preuve par les expérimentations sur le terrain de la faisabilité et les avantages de NDN pour VANET et l'identification des défis restants (Chapitre 3).
- La conception d'un système de mappage entre les noms et l'emplacement des données pour faciliter l'utilisation des techniques de forwarding géographique pour résoudre le problème de récupération de données dans le VANET (Chapitre 4).
- Un mécanisme de forwarding géographique basé sur la topologie routière pour V2V (sur NDN) (Chapitre 4).
- La conception d'un mécanisme de découverte et de sélection capable d'identifier les sources de données disponibles dans de nombreuses régions géographiques et de réagir rapidement aux changements dans les réseaux de véhicules (Chapitre 4).

- La proposition d'un mécanisme de gestion de la mobilité traitant du problème de micro-mobilité (par exemple intra-autonome) des producteurs dans NDN, entièrement distribué et sans compter sur les mises à jour de routage global ou sur l'assistance de composants externes (Chapitre 5).
- La preuve que la technologie actuelle est prête à utiliser des techniques de machine learning basées sur video-processing dans les voitures et les smartphones, afin de recueillir des informations essentielles et en temps réel sur l'environnement. La conception, la mise en œuvre et le déploiement d'un système complet de localisation de stationnement disponible pour voitures, dont le fonctionnement est basé sur edge-computing (Chapitre 6).
- Un nouvel algorithme basé sur l'image-processing qui utilise des estimations de localisation afin d'effectuer une déduplication de les détections d'un même véhicule stationné (Chapitre 6).
- Un nouvel algorithme de localisation pour estimer l'emplacement d'une voiture garée grâce à une image unique, sans intervention ni de l'homme ni de mesure stéréoscopique (Chapitre 6).

Résumé de la thèse Dans le chapitre 2, un bref aperçu de Named Data Networking est présenté, suivi de la description de l'état de l'art des réseaux véhiculaire, en particulier le mécanisme de routage et de forwarding, des solutions de mobilité des nœuds et du paradigme du Fog-computing . Puis, le chapitre 3 présente la description de la première tentative d'exécution de NDN pour VANET (nommé V-NDN), tandis que l'évolution de son système de forwarding est présentée dans le chapitre 4. A suivre, la description de MapMe, une approche sans ancrage au problème de mobilité du producteur (chapitre 5). Enfin, le chapitre 6 décrit la conception et l'évaluation de ParkMaster, un système basé sur le video-processing dans l'edge pour le problème de la recherche de stationnement disponible.

Chapter 2

State of the Art

In this chapter first a description of the Named Data Networking paradigm is presented. Afterwards, the state of the art of VANET is discussed (focusing on routing and forwarding techniques), starting from the IP-based solutions and then continuing to the NDN approaches, more related to the techniques proposed in the rest of this thesis. Remaining withing the ICN domain, the next section discusses the literature about nodes network mobility. Finally, we conclude with the description of Fog Computing.

2.1 Named Data Networking: NDN

2.1.1 Introduction

The Named Data Networking (NDN) architecture, introduced in [71], has its roots in the Content-Centric Networking (CCN) project, publicly presented in 2006 by Van Jacobson. These proposals belong to the family of Information Centric Network architectures, where we can also find the Data Oriented Network Architecture (DONA) [80] and Network of Information (NetInf) [44] solutions.

In the NDN network, each application names the data it wants to fetch, and the network uses these application data names directly to steer packets. Three actors play fundamental roles in this architecture: data producers, consumers, and router/forwarders; they communicate by using application data names directly via two types of packets, Interest and Data (Figure 2.2). A producer names its own data, a consumer sends an *Interest* packet to request a specific piece of *named data*, routers forward Interests toward data producers based on the name carried by the Interest and keep track of all the pending Interests. Each NDN node maintains three major data structures: Content Store (CS), Pending

Interest Table (PIT), and Forwarding Information Base (FIB). The CS caches *data packets* received, which can be potentially useful to satisfy future *Interest* packets. The PIT stores *Interests* that have been forwarded and waiting for matching *Data* packets to return. The FIB is similar to IP router's forwarding table and is maintained by a name-based routing protocol. There is also a *strategy module* that consults FIB in making Interest forwarding decisions. This module is the core part of the forwarder, as it implements the different policies for Interest transmission, for instance in case of multiple next-hop available, in case of failures of previous Interest transmission etc.

When an Interest reaches a node (producer or router cache) with a matching *Data* packet — the Data carries the same name of the Interest, plus some optional name components appended e.g. data name "/home/temperature/13:00" matches the Interest for "/home/temperature/" —, the Data packet follows the PIT entries left by the Interest to get back to the consumer. For each arriving Data packet, a router finds the entry in the PIT that matches the *data name* and forwards the data to all downstream interfaces listed in the PIT entry. It then removes that PIT entry, and caches the Data in the CS (see Figure 2.1). Note

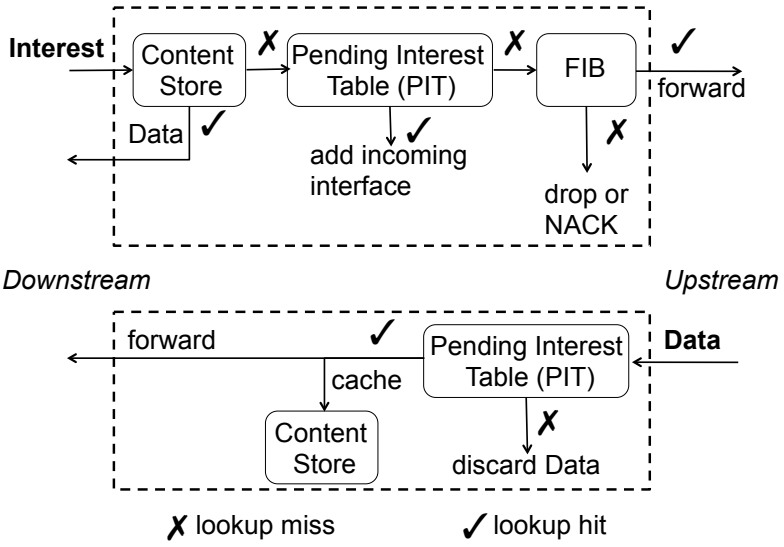


FIGURE 2.1: Forwarding Process in an NDN node

that neither Interest nor Data packets carry IP addresses; Interests are forwarded toward data sources based on the names they carry; Data packets return based on the PIT information set up by the Interests at each hop. The names used in communication are independent from which interface one wants to use, or from whichever nodes the data may come from. Data security and authenticity is embedded by design in the architecture: a producer can encrypt the data if needed and can append a cryptographic signature to the Data packet, binding the name with the content. Consumers (or forwarding node) can verify the

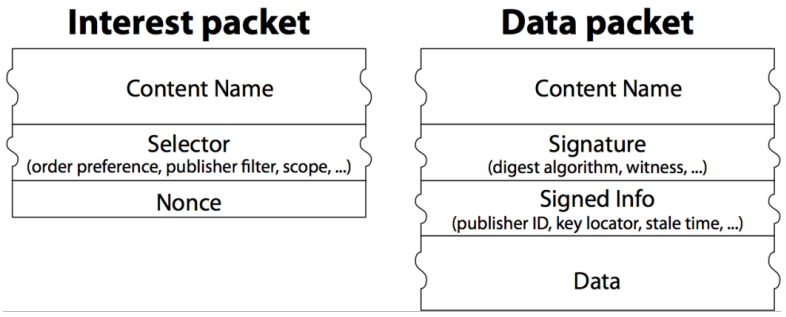


FIGURE 2.2: NDN Interest and Data packets [71]

authenticity of a Data packet by checking its signature, no matter from where the packet is coming from, the original producer or a cache in the network.

2.1.2 Design

Naming The entire NDN architecture is based on naming data and fetching it only based on its name. Thus, the names — or prefixes — play a fundamental role in NDN. By design, names have a hierarchical structure, they are made of multiple components (whose boundaries are indicated by "/" in the text representation). Such a structure allows to define a relation between parts of the same hierarchy and enables prefix aggregation, which facilitates routing scalability. For instance, the prefix "/ndn/NewYorkTimes/" stored in a router may allow the node to properly forward all the Interest for the New York Times newspaper i.e. "/ndn/NewYorkTimes/01May2017/Sport/" for the Sport page published the first of May 2017, and "/ndn/NewYorkTimes/02April2017/Economy/" for the Economy section published in April 2017 etc.

In addition to routing scalability, name hierarchy, together with conventions between producer and consumer, allow the definition of relations among parts of the same data at the application level. For instance, in the case of a video with multiple versions, composed by several chunks (or segments), a name component may be used to specify the version, while the segment may be indicated by the next component. For example, in order to get the *second* segment of the version *one* of *video A* on Youtube a consumer would have to issue Interests for "/Youtube/videoA/1/2", followed by "/Youtube/videoA/1/3", "/Youtube/videoA/1/4" etc. for the next segments of the video.

It must be noticed that such a convention is known by the applications only. Routers in the network are not aware and don't need to be. These nodes only know how to separate components of the same prefix and apply prefix matching (string matching) to process and forward packets.

As already stated, a piece of information is identified by a name, which is used by the network to retrieve the correspondent Data packet. However, this doesn't mean prefixes must uniquely identify a Data packet in the entire network. Indeed, while this may be true for all the content available globally, for local data the uniqueness of the prefix is only local, and depends on the context i.e. if I'm in my house and I'm broadcasting an Interest for "/livingroom/temperature", the request refers to the temperature of my living room. The same prefix in a different house refers to the temperature of a different living room.

While guidelines for data-naming have been set in place already, several questions are still unanswered and the name design can be still considered an open research problem. Testbeds, pilot experiments on the fields and the design of new NDN-based applications drive the efforts of the entire NDN community towards a better naming design.

Security In contrast with IP, where security is a function of where or how the data is obtained, NDN takes a different approach and secures the data itself. By design, trust and protection become properties of the content instead of being related to the channel over which the data travels. Applications are indeed forced to sign each piece of data, together with its name, creating a secure bound between them. Thanks to this bound, given a data packet and producer's public key, any node in the network can (1) verify its integrity (is the content intact and complete?); (2) verify its pertinence, which is explicit in the name, and, (3) validate data provenance (based on the key and the "signed info" packet field – Figure 2.2). Embodying security in content reduces the trust requirements on the network and, in contrast with IP, allows the applications to retrieve "trusted" data also from nodes different from the original source. Keys are content in NDN, thus, as for normal content, producers name keys and use those names to indicate which public key is needed to verify the signature of the produced data packet. As for normal data packet then, consumers, in addition to use the signature to verify and authenticate the packet, match data and key names to verify whether the key is authorized or not to sign that packet. In order for producers and consumers to select and retrieve the proper key to use for a specific data packet, a set of trust rules and schemas have been introduced in [175].

Finally, in case of private information, content (or even names) may be encrypted. Such an encryption is completely transparent to the network, and the decryption keys are distributed as standard data packets.

Forwarding and Routing Interest are forwarded based on names, applying longest prefix matching against FIB. Using the previous New York Time example for instance, the "/ndn/NewYorkTimes/" rule indicates the next hop where to forward Interest for any content published under the "/ndn/NewYorkTimes/" name e.g. Interests for "/ndn/NewYorkTimes/02April2017/Economy/page1" or "/ndn/NewYorkTimes/01May2017/Sport/" would match the aforementioned entry in the FIB. However, in case of a FIB storing also a rule for "/ndn/NewYorkTimes/02April2017/Economy/", the first Interest ("/ndn/NewYorkTimes/02April2017/Economy/page1") would now be forwarded based on the second rule, due to longest prefix match.

Using names to forward and route packets eliminates some of the well-known issues of IP. First of all, names are independent of the network interface used for the communication, thus multi-homing and network mobility don't represent an unsurmountable issue anymore — changing the IP address in use doesn't break the communication, the names are still the same and the Data packet simply follow the reverse path taken by the Interest.

In addition, in contrast with the IP addresses, the names domain is virtually unbounded, thus the address exhaustion problem is no longer present in NDN. Furthermore, since packet forwarding is prefix-based, the addresses of the endpoints don't need to be exposed — NAT traversal doesn't constitute a problem.

While at the forwarding level NDN routers do longest prefix match against the FIB to forward Interest, at the routing level several NDN-based routing protocols have been designed. In particular, NLSR [65] (a Link State Routing protocol) and Hyperpolic routing [91].

Data Plane While the Interests are forwarded based on FIB, Data packets follow the inverse-path of the request — they are forwarded based on the content of PIT. Whenever a node forwards an Interest, it records its prefix, a nonce (a random number carried by the Interest) and the incoming face in the Pending Interest Table (PIT). Such an entry is removed only when a timeout occurs or when a matching Data packet is received. In the latter case, the packet is forwarded on all the incoming faces specified by the PIT entries matching the data name — all the faces where the Interests that requested the Data packet are coming from.

In addition to Data packet forwarding, the PIT serves to avoid loops in the network: whenever a node receives an Interest already present in its PIT (same nonce), it drops the packet, stopping the loop. This loop-free nature of NDN

relieves the forwarding strategy and the routing from any burden in loop avoidance i.e. the PIT will stop the loop, thus the node is free to send Interests over any or all available links. Furthermore, in case of PIT's timeout, the PIT can be used to detect requests' failures (which can be caused by congestion, packet loss, issues with the producer etc.) at the packet level, with a round-trip-time delay. Finally, by recording the incoming faces of the Interests, the PIT naturally supports multicast delivery: in case of multiple requests for the same data (with different nonce — different requester), the Data packet is forwarded over all the faces listed by the PIT entry.

Caching In the NDN architecture each node has a Content Store, a cache to store Data packet in the event that they may be useful again to satisfy other Interests. Whenever a node receives a request, it checks the CS and in case of prefix match, satisfies the request by sending the Data packet to the consumer using the incoming face of the Interest. In contrast with IP-based solutions, which can also have in-network caches, in NDN a Data packet is identified by its name and thus can be re-used multiple times to satisfy different Interests at different moments — the Data packet is not bound to the specific consumer-producer communication that caused its transmission, but it is an independent entity, with its own meaning, given by its name. Furthermore, as already discussed in the security section, each data packet is signed by the producer, which allows, even in case of content coming from CS, verification of data integrity and provenance by any node in the network.

2.1.3 NDN project current status

The efforts of the entire NDN community has been directed towards the design and development of an open platform available to researcher for the design, testing and deployment of new NDN-based solutions at large scale. The core component of such a platform is the NDN Forwarding Daemon (NFD), developed by a joint effort of several institutions and developers [8]. Such a daemon runs on the NDN testbed shown in Figure 2.3, which currently[1] includes 34 nodes across four continents.

[1] June 2017.

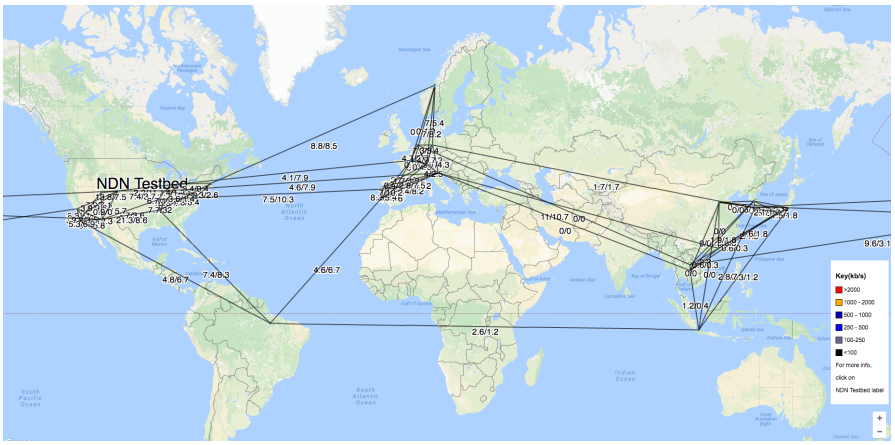


FIGURE 2.3: NDN testbed

2.2 Vehicular Networks

2.2.1 IP-based solutions

VANETs have been studied for decades and plenty of works have been done, in particular on routing and forwarding. Reusing the classification of [94], it's possible to categorize the state of the art in Ad Hoc, position-based, cluster-based, broadcast and geocast routing approaches.

The first protocols that need to be mentioned in the **Ad hoc** routing category are AODV (Ad hoc On-demand Distance Vector) [123] and DSR (Dynamic Source Routing) [75]. While these two protocols have been designed specifically for MANET and it has been proven that they have limitations in highly dynamic scenarios (like VANET), some modifications to address those scenarios have been proposed in the literature. For instance, PRAODV and PRAODVM [111] are both modifications to AODV that use mobility predictions in the routing decision process. The first predicts links duration based on cars speed and location and tries to find new alternative paths before the estimated link duration is over — thus before the link actually breaks. PRAODVM's approach is similar, with the only difference in the route selection policy: instead of picking the shortest path, PRAODVM selects the more stable, the one with the maximum predicted lifetime. The main limitation with both approaches consists in the mobility prediction, whose accuracy highly affects the performance of these routing protocols. Another modification of AODV is proposed in [117], where an approach similar to LAR [79] is used: a zone of relevance (ZOR) is specified, and the route requests defined in AODV are forwarded only within the desired ZOR.

In the **cluster-based** category fall all the protocols proposing a virtual network built by clusters of nodes. Typically, nodes within a cluster communicate among

themselves with direct links while for inter-cluster communications and cluster coordination a head-node for each cluster is selected. Similarly to the Ad Hoc routing case, also for the cluster-based category the protocols designed for MANET encounters limitations, due to the high dynamism of the VANET, different user behavior and different mobility characteristics (i.e. cars' movements are constrained by roads). Among the cluster-based protocols designed specifically for VANET, we have COIN [30], which selects the head of each cluster taking into considerations vehicular dynamics and drivers behavior. LORA_CBF [130] instead defines different roles for each cluster: the cluster head, the gateway (a vehicle connected to multiple clusters) and cluster members. A greedy routing approach is used to route a packet towards its destination. However, if the latter is unknown, the head of the cluster and the gateways take care of finding its location, by sending location requests. Finally, [96] proposed a cluster-based approach for data dissemination in VANET. In contrast with Ad Hoc routing protocols, cluster-based approaches usually have good scalability for large networks. However, maintaining a cluster may become an issue when cars come and go.

In the **broadcast** category we can find protocols used for data-dissemination that requires multi-hop communication, like, for instance, the delivery of road condition messages, advertisements, traffic condition etc. In this category, in addition to the simple flooding approach, we have BROADCOMM [45], designed specifically for highways, that divides the road in virtual cells and, based on cars' positions, assigns the role of cell reflector to some specific vehicles. Each reflector handles the forwarding of packets coming from different cells, reducing therefore the overhead. UMB [81] implements a forwarding node selection process based on distance to the sender and a "request-to-broadcast clear-to-broadcast" process similar to RTS/CTS. Finally, V-TRADE and HV-TRADE [143] divides neighbors in different forwarding groups (based on position and movements). For each group, only a subset of nodes are enabled to forward packets.

The **Position-based** category includes the protocols that uses information like road topology and traffic status in the routing process. In this category, although not IP-based, we can found the protocol discussed in chapter 4. Among the IP-based routing protocols instead, we can find GPSR (Greedy Perimeter Stateless Routing) [76], one of the most known position based routing, which mixes greedy routing with face routing when the first fails to deliver the packet to the destination. GSR (Geographic Source Routing) [99] integrates streets map with the routing process: given the location of the destination, the paper proposes to use Dijkstra's shortest path to find the list of road-intersections that need to be crossed to reach the destination. GPCR (Greedy Perimeter

Coordinator Routing) [100] picks cars at the road intersections as "coordinator". In order to deliver a packet, the protocol uses a greedy approach, where only the coordinators can forward the packets. Whenever the greedy approach fails, a repair strategy is put in place in order to exit the local minimum. Similarly to GSR, A-STAR [97] computes the list of intersection that a packet should cross to reach its destination. In order to compute such a path, A-STAR takes into consideration the traffic level on each street. In addition, whenever a local minimum occurs, the area is momentarily marked as unavailable in order to avoid other packets to fall in the same local minimum. GpsrJ+ (Gpsr Junction+) [88] aims at improving GPCR performance by using a digital street map in order to identify vehicles at a junction and by using information about the next hop after the coordinator to predict forwarding directions (whenever the direction doesn't change, the packet can be forwarded directly by the next hop without passing by the coordinator). Position-based Directional Vehicular Routing(PDVR) [147] tries to maintain a stable route based on vehicles' position and direction, and the position of the destination: the next hop should be traveling in the same direction as the source vehicle and lie in the direction of the destination. In the improved greedy traffic aware routing protocol (GyTAR) [72] the selection of the roads a packet should traverse takes into consideration the distance to the destination and the traffic density. In order to obtain the latter, each road is divided into small cells, with one group leader, which takes care of gathering and exchanging traffic level information. Topology-assisted Geo-Opportunistic Routing (TO-GO) [90] improves GpsrJ+ with opportunistic forwarding. TO-GO identifies a forwarding set between sender and anchor node, and the nodes compute a waiting time before forwarding a packet based on their distance to the anchor. GeoDTN+Nav [41] combines different approaches, such as greedy, perimeter and DTN: packets are first forwarded in greedy mode, and pass to recovery mode if a local maximum is reached. Whenever the recovery mode fails, the third, final mode, is entered, the DTN mode. Finally, Landmark Overlays for Urban Vehicular Routing Environments (LOUVRE) [89] defines an overlay network on top of the urban topology, where junctions are nodes of the network and a link exists only if the estimated traffic density on the corresponding road can guarantee multi-hop network connectivity among the two intersections. The resulting network is used to computes routes.

The **geocast** includes all the position-based approaches whose goal is to deliver messages to all the nodes in a specific area. In this category we have [35], that addresses the broadcast storm problem by introducing a distance to the previous hop based waiting time before forwarding a packet: longer is the distance between a node and the sender, smaller is the waiting time before forwarding the packet. [23] takes a similar approach in order to propagate alarm messages in

case of emergency. The use of caching is instead proposed by [103], where a small cache is added to the routing layer, allowing packets that cannot be delivered instantly to be momentarily cached.

2.2.2 NDN-based solutions

The literature on VANET doesn't stop at IP-based solutions, but it extends also to the Information Centric Network family of architectures.

First, in 2010, [25] proposed a generic network framework dubbed "Information Centric Networking on Wheel" with an overlay network to guarantee the coexistence with IP protocols. The paper argues that in-network and decentralized data replication would be beneficial for vehicular applications and focuses on the spatial and temporal scope of the carried data in such an environment.

In the same year [162] explored the naming scheme for vehicular application and has a first and preliminary discussion about the benefits of the ICN (specifically, NDN) approach in VANET. The paper designs a data collection system to collect information from cars i.e. for monitoring or alert purpose, where RSUs are used to broadcast Interests and retrieve information produced by vehicles.

Afterwards, in 2011, [20] applied the CCN framework to evaluate safety data dissemination in VANET, when cars are equipped with multiple radio interfaces. The paper introduces a new type of unsolicited packet for emergency messages and a prioritization mechanism to speed-up the transmission of certain type of packets.

In 2012 [17] presented CRoWN, a content centric approach implemented on top of the IEEE 802.11p protocol. The paper proposes a two-phase data recovery: first flooding is used to retrieve the first chunk and "discover" the path to the producer. Later on, Interests for the next chunks are forwarded only along this path — by cars that have seen the first chunk — implementing a hop-based distance to the producer mechanism to decide which nodes should forward the requests. [165] evaluates the V2V direct communication, use cases and requirements and analyze the naming design from an application point of view. [163] shows that when applications use location-based data, encoding geolocations into names may be beneficial for the Interest forwarding process. It also proposes to randomize packet forwarding based on distance from previous hop to avoid collision and utilize implicit acknowledgment to abort transmission (nearby car forwarding the same packet), focusing on the highway scenario. [16] presents a preliminary evaluation of the Content Centric approach to VANET

(CCVN), showing its advantages over TCP/IP based solutions. [145] applies network coding techniques to improve content delivery and dissemination, exploiting redundant path. [177] uses bloom filters (BFR) to pro-actively advertise the name prefixes to facilitate the retrieval of popular content. The network is divided in a hierarchy of clusters and bloom filters are advertised by the heads of the clusters.

In 2013, with [61] started my work on NDN: we sketched the first implementation of NDN for VANET on real hardware, including both V2V and V2I aspects in its (preliminary) evaluation. [19] enhance authors' previous work about CCVN [16]. The paper proposes deferring transmission timer to exploit broadcast communication while limiting the broadcast storm problem. In [176] cars exchange encounter information and the Interest is flooded only when the location of the content is unknown. Otherwise, geo-routing techniques are used to retrieve the data. Finally, [18] addresses two different problematics in VANET, packet segmentation and reliability. Authors evaluate the best chunk size for Data packets and propose a consumer-driven retransmission policy whenever an Interest remain unsatisfied.

In 2014, the paper "VANET via named data networking" [59] extended [61]'s designs, implementation and evaluation of NDN for V2V and V2I urban scenarios. This paper constitutes the beginning of my work on NDN for VANET during my PhD and it is discussed in chapter 3.

Later on, [11] proposes a periodic exchange of information among neighboring nodes, the list of Interest satisfied by the node (RLS), in order to built a list of Interest satisfied by neighbors (NSL). This list is then used to decide which node should forward the Interests. [178] extends previous work [177] and proposes a distributed bloom filter aggregation reliving the heads of the clusters from the need to advertise prefixes bloom filters.

The routing and forwarding problem in content centric based VANET has been the focus of plenty of other works. Among the works which propose geo-based strategies, we can find [98] that support geo-tagged name based information, [28] and [29].

2.3 Producer's mobility

Many efforts have been made to define mobility-management models for IP networks in the last two decades, resulting in a variety of complex, often not implemented, proposals. A good survey of these approaches is RFC 6301 [183].

Likewise, within the ICN family, different approaches to mobility–management have been presented [155]. In DONA [80] mobile publishers unregister and re–register their information at each handoff to the hierarchy of resolution handlers. Such an update process, however, may incur in a non-negligible messaging overhead to eliminate stale registration across the network [171]. Similarly, NetInf [10] and JUNO [153] report network mobility events to a resolution service, which may incur in network load in case of high mobility [154]. PURSUIT [50] instead uses a rendezvous system to handle network mobility, which requires notification to the topology manager at each handoff and, in some cases, the re-computation of the forwarding identifier used to compute the path to the producer, affecting the handoff delay [171, 154]. Finally MobilityFirst [135] uses a global name resolution service (GNRS), which is updated when a node changes point of attachment. When facing high-frequency mobility, those so-called Resolution-Based (RB) approaches present a similar trade-off: for every packet the consumer has to resolve the producer's location or use stale information and run the risk to reach an old position, incurring in timeout, or Nack, etc.

Specifically for the CCN/NDN solutions, several surveys of mobility-management approaches can be found [180, 48]. In [180] for instance, the authors distinguish three categories of solutions – routing, mapping, and tracing-based – depending on the type of indirection point (also called Rendez-Vous, RV). We build on such classification and extend it to distinguish a fifth class of approaches not relying upon the existence of any fixed anchor point as the RV (Local Routing):

a) **Routing-based (RT)** solutions rely on intra-domain routing, and require updating all routing in the AS after a mobile's movement. Scalability of these solutions is widely recognized as a concern which explains why they are usually ruled out, in particular for CCN/NDN where the name space is even larger than IP.

b) **Resolution-based (RB)** solutions rely on dedicated RV nodes (similar to DNS) which map content names into routable location identifiers. To maintain this mapping updated, the producer signals every movement to the RV node [64, 74, 93, 77, 9, 78]. Once the resolution is performed, packets can be correctly routed from the consumer along the shortest path, with unitary path stretch (defined as the ratio between the realized path length over the shortest path). Requiring explicit resolution, together with a strict separation of names and locators, RB solutions involve a scalable CCN/NDN routing infrastructure able to leverage forwarding hints [64, 74]; however, scalability is achieved at the cost of a large hand-off delay as evaluated e.g. in [77, 48] due to RV update and name resolution. To summarize, RB solutions show good scalability properties

and low stretch in terms of consumer to producer routing path, but result to be unsuitable for frequent mobility and for reactive rerouting of latency-sensitive traffic.

c) **Anchor-based (AB)** proposals are inspired by Mobile IP, and maintain a mapping at network-layer by using a stable home address advertised by a RV node, or anchor. This acts as a relay, forwarding through tunneling both interests to the producer, and data packets coming back. This is for instance the case in [87], where the producer changes its prefix after each movement and then sends an update message to its anchor to notify it of the change. In such context, anchor's placement is critical for the performance of the approach.
MobiCCN [166] uses distributed anchors and selects the closest in a hyperbolic space.

Advantages of this approach are that the consumer does not need to be aware of producer mobility and that it has low signaling overhead because only the anchor has to be updated. It however inherits the drawbacks of Mobile IP – e.g. triangular routing and single point of failure – and others more specific to the CCN/NDN context: potential degradation of caching efficiency, bad integrity verification due to the renaming of content during movement. It also hinders multipath capabilities and limits the robustness to failure and congestion initially offered by the architecture.

c) **Tracing-based (TB)** solutions allow the mobile node to create a hop-by-hop forwarding reverse path from its RV back to itself by propagating and keeping alive traces stored by all involved routers. Forwarding to the new location is enabled without tunneling. Like AB though, this approach assumes that the data is published under a stable RV prefix. Kite [181] introduced this approach and proposed storing traces in the PIT to build a breadcrumb trail which could be followed by crossing consumer interests and thus provide a shortcut towards the producer. While it exploits CCN/NDN data plane features without requiring a separate control infrastructure, Kite involves a large signaling due to keep-alive messages to maintain active traces stored in PITs. The idea of creating a reverse path to a stable home router is also expressed in [62], where the authors propose a similar tracing-based approach, leveraging updates in FIB, rather than in PIT, and sending updates to both RV and previous PoA.

e) **Local Routing (LR)** approaches allow the mobile nodes to advertise their mobility to part of the network without requiring any specific node to act as a RV. In case of handoff, only a small part of the network is aware of the movements and acts to maintain producer reachability. Typically the region of updated nodes is determined based on the current location of the producer,

without the need of predefining nodes in the network to act as RV point between consumer and producer.

These approaches are less common and introduced in CCN/NDN to enhance the reactivity with respect to AB solutions by leveraging CCN/NDN name-based routing. [124] exploits multicast and directs the same Interest to the nearby PoAs of the producer. In [173] and in the *Interest Forwarding* scheme proposed in [77], the mobile producer sends a notification to its current PoA before moving. The PoA starts buffering incoming Interests for the mobile producer until a forwarding update is completed and a new route is built to reach the current location of the producer. Enhancement of such solutions considers handover prediction. Besides the potentially improved delay performance w.r.t. other categories of approaches, some drawbacks can be recognized: buffering of Interests may lead to timeouts for latency-sensitive applications and handover prediction is hard to perform in many cases. [115] instead introduces proxy nodes at the edge of 3G/4G architectures and uses tunnels to forward Interest from the former PoA to the current edge. The solution, however, is specific to cellular network.

Finally, in-network caching and name-based routing techniques also enable a routing-to-replica approach abstracting consumers from producer movements (referred to as data depot in [180]). However, such an approach is not suitable for realtime applications or targeted to unpopular content, which may be replaced in cache due to memory limitations. A study of the advantages for popular items can be found in [74].

2.4 Fog Computing

In 2012 Cisco introduced the term "Fog computing" [32]. This approach extends the concept of cloud and pushes some of the network and computing tasks from the cloud to devices closer to the end-user, towards the "edge of the network".

The initial definition of Fog computing can be synthesized as a network architecture that uses a multitude of end-users' devices or close to the users edge-devices to perform some of the tasks that traditionally the cloud performs, from storage (instead of rely entirely on remote data centers), communication (instead of sending everything to the cloud), sensor management, control and configuration.

However, the term Fog computing doesn't identify one single specific architecture, but a more generic approach of pushing the intelligence closer to

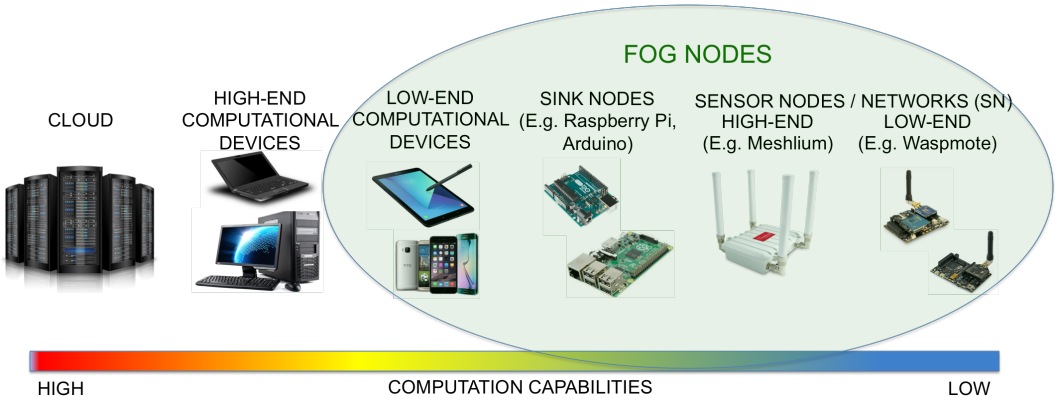


FIGURE 2.4: Fog architecture: Categorizations of IoT devices based on their computational capabilities [120].

the end-user, promoting to use leaf nodes (close to the user) as much as possible, in a collaborative manner if needed, and of course taking into consideration the limitations of those devices (power, computation etc.). [157] tried to make some clarity and expanded the Fog computing concept is such a way: "Fog computing is a scenario where a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralized devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third parties. These tasks can be for supporting basic network functions or new services and applications that run in a sandboxed environment. Users leasing part of their devices to host these services get incentives for doing so".

Instead of sending all the collected data to the cloud, Fog (or Edge [119]) computing suggests to process (at least partially) the data in-loco — closer to where it has been produced — at leaf nodes or at the edges (the so called "edge analytics" approach). Pre-processing data at the edge allows to address some of the weakness of the cloud, to reduce delays and bandwidth requirements, to bring context-awareness to the computation etc.

While there is no strict definition, usually as fog (edge) nodes are considered those devices with some (limited) capabilities to process or store data and to move it across the network. Figure 2.4 shows the classification of edge nodes adopted by [120], which includes small low-end sensor nodes, devices as Raspberry and Arduino boards, and more sophisticated and powerful — but still limited — devices as smartphones and tablet. All of those devices (smart plugs, smart fridge, smart-watch, smart-phone etc.) can become a leaf node and perform "edge analytics".

2.4.1 Use cases

With the proliferation of IoT and smart devices and the diffusion of smart cities, plenty of applications can be found for the Fog architecture. Anywhere large amount of data is produced and needs to be processed (at least partially) locally, the Fog can play an important role and improve efficiency, reactivity, privacy or security. Briefly, here are listed few use cases already described in literature [120]:

- **Smart Agriculture:** projects like Phenonet [132] have already presented case studies about the smart agriculture concept, where sensors are used to constantly monitor and control all the agricultural activities. In such a scenario, context information is crucial and data has to be collected in a timely and location-sensitive manner in order to optimize both agricultural activities and data collection (i.e. data collection frequency based on weather, temperature etc.) [121].
- **Smart Transportation:** this use case may include for instance pollution sensors deployed in public buses, taxis etc., but also infotainment, safety, traffic support, and analytics services for the connected car [32].
- **Smart Health and Well-Being:** this is the case of wearable devices designed to monitor the health status of the user (i.e. hearth rate, blood pressure etc.), where the edge that process the raw data and that can take the immediate decisions may become, for instance, the user's phone. An example of such a case can be found in [179]. The authors propose to use the fog computing architecture to develop a brain computer interaction application and pre-process EEG data at the edge.
- **Smart Waste Management:** sensors can be deployed to optimize garbage collection strategies (i.e. on the garbage bin) or to monitor the waste management process without human intervention. In this case the edge can play the role of data collector and aggregator and upload to the cloud only a smaller version of the data set.
- In addition, other uses cases are represented by **Smart Retail Store Automation** (i.e. advertisement); **Smart Power Grid** (i.e. optimization, fast recovery after disruptions); **Smart Greenhouse Gases Control**; **Smart Water Management** etc.

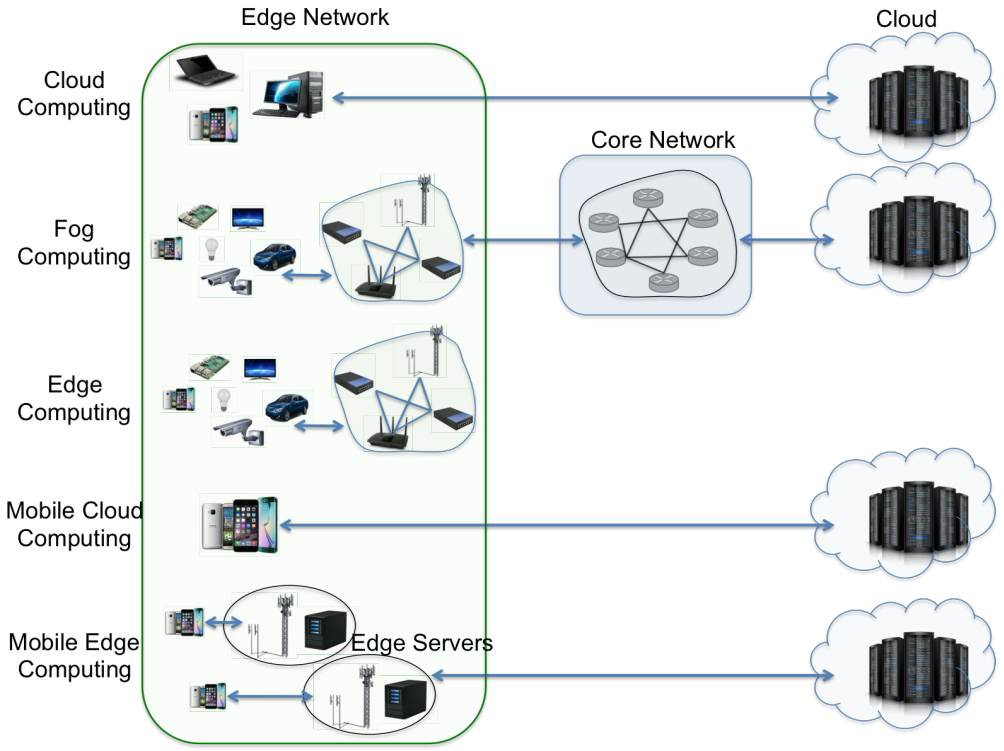


FIGURE 2.5: Computation domain of Cloud, Fog, Edge,Mobile Cloud and Mobile Edge computing [102].

2.4.2 Related work

Similar concepts to the Fog computing are present in the literature [102, 128]. Here the most relevant are briefly discussed (Figure 2.5).

Edge Computing The term "Edge computing" has been proposed by [54]. The main concept is, as for Fog computing, to bring computation tasks closer to the source of the data. This architecture enables data processing at the edge of the network [158], on devices like end-devices (i.e. smartphones, IoT devices), edge devices (i.e. border routers, base stations, wireless access points etc.) and edge server. A federation of edge-centric distributed services is deployed across small "data centers" at the edge of the network, which may collaborate with each other in a peer-to-peer fashion to complete the task. By handling the computation at the edge — closer to the source of data — delays for computational services are usually lower. In contrast with Fog computing, however, generally Edge computing focuses only at the edge of the network, without including cloud services [138].

Mobile Edge Computing The concept of Mobile Edge Computing (MEC) was initially introduced in 2013 by IBM and Nokia [110], as a platform to run applications and services within a mobile base station. Afterwards, in 2014, ETSI started the Industry Specification Group (ISG) for Mobile-Edge Computing [109], and extended the Mobile Edge Computing concept to an architecture that provides cloud-computing capabilities at the edge of the mobile network, including thus aspects like application migration, interoperability, etc., which were not present in the definition of 2013. Similarly to the aforementioned architectures, the benefits of performing cloud-tasks at the edge of the networks are low delays, bandwidth savings and local awareness.

Mobile Cloud Computing The Mobile Cloud Computing (MCC), in contrast with the previous approaches, has its main characteristics in "delegating" the tasks pertinent to the mobile device to the more resourceful cloud. In the first definition of MCC, introduced in 2009 by [13], the only place for computation was the centralized clouds. However, later versions included devices located at the edge as suitable places for performing the mobile device's tasks [24]. The edge provides an execution platform located close to the mobile devices to perform several type of tasks, with, as in the previous cases, the benefits of lower latency and access to context information.

Within this category falls the concept of **Cloudlet** [131], which refers to a smaller version of the cloud architecture, located close to the mobile user (for instance inside shops, buildings, cars etc.), that allows to load small Virtual Machine overlay to perform the requested tasks.

Chapter 3

V-NDN: NDN in Vehicular Networks

3.1 V-NDN: A proof of concept

In the near future, a car will be equipped with a variety of wireless interfaces such as 3G/LTE, WiMAX, WiFi, or DSRC/WAVE (for instance the US DOT is proposing wireless V2V technology as mandatory in newly manufactured cars [4]). Our vision is to enable vehicles to communicate with each other and with the infrastructure over any and all physical communication channels, as soon as any channel comes into existence and as long as it is available. Although over the years many research papers have been published for automotive research, in reality today's vehicles are mainly connected through cellular networks to centralized servers only. Automotive research such as Ad Hoc networking and delay tolerant networking are still far from completion and less likely to deploy. We believe the root cause of this insoluble problem in networking vehicles is IP's communication model, where IP creates its own name space, the IP address space, assigns IP addresses to every communicating end point, and then encapsulates each piece of application data into an IP packet. This whole process insulates applications from data delivery layer.

In this chapter we apply the design of Named Data Networking (NDN) to address VANET challenges. We show that naming data decouples communication from specific interfaces and endpoints, enabling a car to utilize any available interfaces and fetch data from any other nodes as soon as physical connectivity comes into existence. As a proof of concept, we designed and

implemented a prototype of Vehicular NDN, V-NDN [59][1], and demonstrated its utility through real experimentation. We also used simulations to explore V-NDN feasibility at large scale. The contributions of this chapter can be summarized as follows: (1) we articulated the new NDN functional requirements for VANET and sketched out initial solutions (Section 3.1); (2) we developed a prototype V-NDN implementation (Section 3.2.1); (3) we conducted experimentation via both demonstration (Section 3.3) and simulation (Section 3.4); and (4) we identified remaining challenges in rolling out V-NDN (Section 3.5). Those challenges will then be addresses in chapter 4.

3.2 Design and Implementation

Vehicular networking possesses two fundamental characteristics: *ad hoc, intermittent* connectivity, and the ability to physically transport data. In a V-NDN network, a car may play any of the four roles: data consumer, data producer, forwarder when it is connected to either infrastructure or other vehicles, and "data mule" when it carries data across distance while having no connectivity to anyone else. Different from other types of mobile devices, vehicles have limited concern with computation/storage capacity or power supply.

NDN is a great enabler to vehicle networking by removing the constraints in TCP/IP protocol stack, however several modifications to the baseline NDN operations are necessary for VANET environment. First, since all communications are over wireless channels, one should take full advantage from wireless broadcast nature. Instead of only accepting data with matching entries in PIT, a vehicle may want to cache all received data regardless of whether it has a matching PIT entry or whether it needs the data itself. Since a car can have much bigger data storage than mobile phones, this opportunistic caching strategy can be advantageous in facilitating rapid data dissemination in highly dynamic environment.

Second, Data packets can be carried by running cars even when they have no network connectivity. Indeed data can move away from the producer's location either by requests or by car movements. When a car responds to an Interest with data, this data reply, via wireless broadcast channel, can spread to neighboring cars and be cached by all the receivers. When these cars physically move around, they serve as data mules carrying the content to wider area. Having large number of mules enlarges data spreading areas and increases rendezvous

[1]The work on V-NDN has begun before starting the Ph.D. program, while being a visiting researcher at UCLA, and has been concluded later on while being a Ph.D. candidate at UPMC.

opportunity between consumers looking for a specific piece of data and mules carrying a copy of it.

Finally, the high dynamics of connectivity among moving vehicles makes it difficult, if not completely infeasible, to run a routing protocol to build the FIB. Therefore V-NDN must develop other means to guide Interest packet forwarding.

In this section we describe the V-NDN implementation and the modifications to NDN needed to accommodate VANET specific features.

3.2.1 Implementation

We developed a V-NDN prototype under linux Ubuntu 12.04[2]. The implementation overview is depicted in Fig. 3.1.

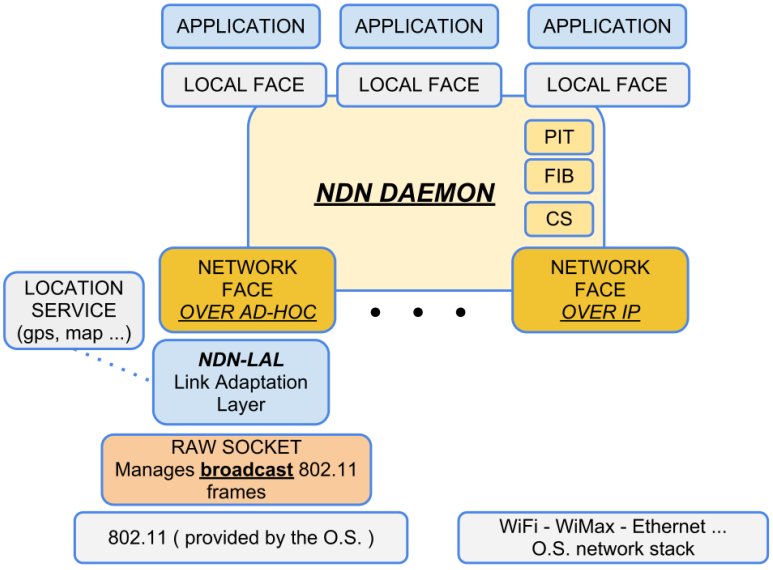


FIGURE 3.1: V-NDN implementation framework

NDN Daemon It provides the core NDN capabilities by maintaining the key data structures of CS, PIT and FIB and taking care of name prefix-matching and packet forwarding decisions. If the node is equipped with multiple network interfaces, the current implementation takes a simple approach of forwarding each Interest to all the interfaces that are available at the time, therefore the FIB is not used in Interest forwarding at this time. The exploration of forwarding

[2]The source code is available at https://github.com/named-data/vndn. There are no kernel dependencies and the software is expected to run smoothly under any Linux distribution.

strategy design [172] for VANET is part of our next work. Finally, the CS caches all Data packets overheard on the wireless channel (solicited or not)[3].

NDN Local Faces These are interfaces between the Applications and the NDN daemon. They support application registration (an application registers with the local FIB the name prefix of the data it produces), Interest request, and content delivery.

NDN Network Faces These Faces provide the specific adaptation functions coupled with the technology used in the communication. We use IEEE802.11 wireless technology in Ad Hoc mode for V2V and provide the interface with the *Link Adaptation Layer* which supports WiFi broadcast (see below). We use several wireless technologies for V2I including WiMax, 3G, and WiFi mesh networks. For 3G connectivity, the NDN Network Face provides the adaptation to an IP tunnel between the mobile and some NDN node in the core network.

Link Adaptation Layer (LAL) LAL is conceptually layer 2.5, designed to efficiently take advantage of specific layer 2 technologies. LAL sends all packets as L2 broadcast, using IEEE802.11[4] frame to carry NDN packets directly. However IEEE802.11 broadcast support is practically nonexistent, leaving a number of tasks to be surrogated by LAL [67][66]. We discuss our solution to this problem below.

Location Service It provides reverse-geocoding capabilities, as well as high level functions on distance and heading, to the LAL. The Link Adaptation Layer uses them to geographically scope the communication. Furthermore, some applications may choose to encode location information in data names when the content concerns a specific area, as traffic or parking information. The location might be used to facilitate NDN Interest forwarding [163].

Compared with the current implementation of the NDN Forwarding Daemon (NFD) [8] (realized after V-NDN), the NDN daemon shown in Figure 3.1 corresponds to the "forwarding" module of NFD, while faces, PIT, CS and FIB in V-NDN match the corresponding NFD version.

[3]The current implementation keeps CS in main memory with a size limit of 10GB; cached items have no expiration time other than being swapped out. The next version will employ disk-based CS and support smart caching policies.

[4]We intentionally refer to the entire 802.11 family instead of the specific protocol because the differences between the various versions don't affect LAL.

3.2.2 Enhancing WiFi Broadcast for V2V communications

We use IEEE802.11 broadcast for all V2V communications, which requires additional support to provide reliability. Indeed, the current IEEE802.11 standards provide neither collision prevention for broadcast transmission, nor collision detection/recovery mechanism. Furthermore, WiFi broadcast can suffer high losses due to collision, which can be further exacerbated by the nature of vehicular networks that feature relatively short link durations and fast changing topologies [129].

Thus, to enable efficient and resilient broadcast transmission, which in turn enables opportunistic forwarding and caching in V-NDN, we developed a simple set of mechanisms to provide WiFi broadcast support in VANET communications. Our WiFi broadcast support is directly coupled with our packet forwarding algorithm, as described below. We assume that each vehicle is equipped with GPS and a digital map. We use a simple greedy forwarding strategy to spread NDN Interest packets in *all directions* in the following way. Each Interest packet I carries the location information of its sender S. When I is received by multiple surrounding vehicles, the one that is furthest from S should forward I; the other receivers simply do nothing. Furthermore, S needs to know whether I is being received and further forwarded, otherwise S needs to retransmit I.

We implemented the above mechanisms in the Link Adaptation Layer. Equipped with GPS data, LAL at each node N computes the distance between the sender and itself, then sets a random wait timer, the *Forwarding Timer*, based on this distance value: the shorter the distance, the longer the wait. During this wait time, if N hears the forwarding of I by another node F, N uses reverse geocoding techniques to locate F on the map and identify the road segment where F resides. This transmission is considered by N as an implicit (partial) acknowledgment. If N hears implicit (partial) acknowledgment from each of the streets stemming from where N is located (except the street where I was initially coming from, i.e. the location of S), it considers the packet as completely acknowledged and cancels its own Forwarding Timer; otherwise when the timer expires, N will forward the packet. Similarly, the forwarded packet can also be heard by sender S, and if S doesn't hear implicit acknowledgment from all the streets stemming from its location, it will retransmit the packet. All retries are upper-bounded by a preset limit n, the packet is dropped after n unsuccessful attempts.

This weighted random wait scheme statistically allows the node furthest away from a packet sender to forward the packet, resulting in fast packet propagation.

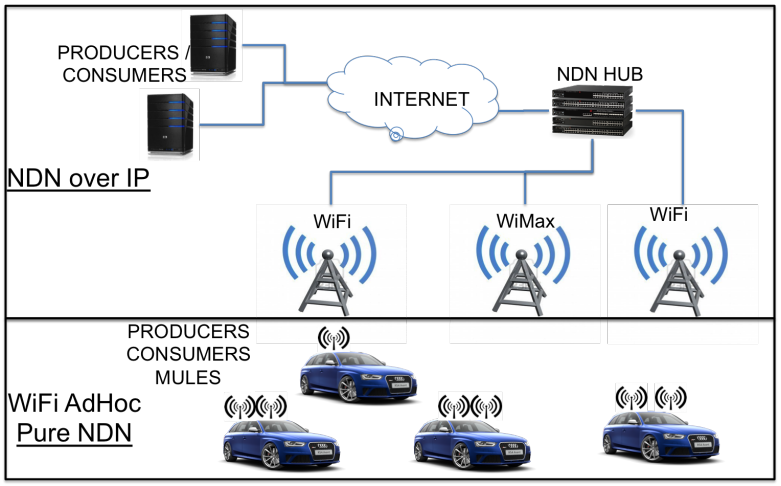


FIGURE 3.2: V-NDN: Testbed network configuration.

LAL combines two different components to compute the timer: a deterministic component, $\frac{1}{D(sender,receiver)}$, where D distance is computed using the location service, and a small random component used to randomize the transmission time. The first component favors cars further away the last sender, the second one reduces the collision probability among the nodes with same distance from the last-hop.

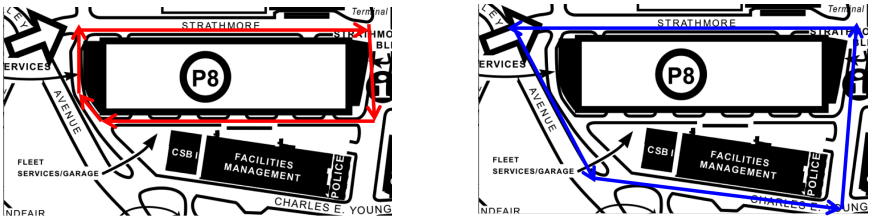
To limit the excessive spreading of Interests on the vehicular network, we add a hop-counter to the LAL header which is decreased at each hop. An Interest is dropped when its hop count exceeds a preset limit (the experimentations described below use a limit of 5 hops).

3.3 Demonstration

We implemented a prototype of V-NDN as proof of concept and tested it using the UCLA Vehicular Testbed. The experiments involved multiple vehicles (up to 10 cars). We implemented two simple vehicle applications over NDN: **Info-Traffic** and **Road-Photo**. The first application emulates traffic information requests for a specific area; the area is encoded in the name carried in Interest packets. Instead of coordinates, we name intersections and streets stemming from that intersections, e.g. /traffic/westwood-at-strathmore/ refers to traffic information from the area surrounding the Westwood-Strathmore intersection. A car that has been or is currently in the location will respond to the Interest with the proper traffic information. The road-photo application emulates photo requests for a specific area, and any vehicle that has been in that area recently and has taken a photo may respond.

The experiments were designed to investigate V-NDN behavior in the following communication scenarios typical for the vehicular applications domain (Figure 3.2):

- **V2V**: Vehicles exchange Interests and Data packets over WiFi (Ad Hoc), the requested data is relatively local (e.g. a few hops away).
- **I2V**: The consumer node resides on the wired network (e.g. traffic control center server) and data producer nodes are the vehicles that are close to, or have been recently close to, a point of interest and have information about the traffic.
- **V2I**: The consumer is on the vehicle while the producer is connected to the Internet through wired or wireless network, e.g. a vehicle wants to retrieve data about the traffic of an area from a centralized server.
- **Network disruption**: Vehicular networks are prone to disruption, the link duration is relatively short and the topology is continuously changing, resulting in dynamic network partitions [129].
- **In-network storage**: One of the NDN advantages is enabling caching in the network. This feature turns out to be essential for communication in VANETs.



(a) V-NDN Driving Route, Clock Wise one block around Parking Lot 8

(b) V-NDN Driving Route, Clock Wise two blocks around Parking Lot 8

FIGURE 3.3: Mobility Patterns

3.3.1 Field Experiments

We performed a number of field experiments for both applications varying mobility patterns and the communication types involved (i.e. V2V, V2I/I2V, V2V2I).

Still These experiments were performed on the rooftop of UCLA parking structure P8 (265m by 76m) with *no mobility* and cars in line.

Platooning These experiments were performed on the rooftop of P8 with very basic *slow urban platooning*.

Moving around campus These experiments were performed by driving around P8. The pool of 10 vehicles was divided in two groups of 6 and 4 cars each. The smaller group of vehicle ran clockwise around P8 as shown in Fig. 3.3(a); the larger group ran counterclockwise and covered a larger road block which also includes P8 (see Fig. 3.3(b)). Car speeds ranged from $6.3_{m/s}$ to $21.2_{m/s}$ (i.e. ~14 to 47mph). This mobility pattern allows vehicles traveling in opposite direction to meet, but prevents continuos connectivities between the two groups. Traffic lights and pedestrians created dynamic partitioning in each group, as well as voids as they happen during regular course of campus traffic [129].

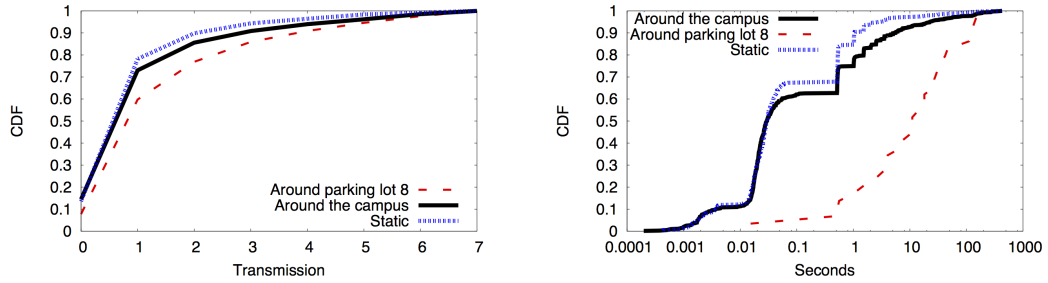
3.3.1.1 Hardware/Software Setup

V-NDN was installed in low-cost netbooks the *Asus EeePc 1011CX* powered by an Intel Atom N80 at 1.6GHz; each node was retro-fitted by us with a MIMO-capable WiFi Card, the *Unex DNXA-92*, based on the Atheros AR9280. All the experiments were performed using the 2.4GHz band (Channel 1). Two nodes out of the 10 were equipped with a second WiFi interface operating in Infrastructure mode (we used an Ubiquity Networks SR71USB); two other nodes were equipped with a WiMax usb based interface. We used an off-the-shelf Ubuntu Desktop 12.04 as operating system and V-NDN was installed in user space, no kernel hacks are required. V-NDN transmits broadcast packets in the WiFi Interfaces operating in Ad Hoc mode, and tunnels NDN traffic over IP for the other interfaces (see Fig.3.2). A V-NDN hub is connected at the other tunnel endpoint and performs the appropriate forwarding tasks.

3.3.2 Preliminary Results

3.3.2.1 V2V experiments

Here we focus on the effectiveness and cost of our LAL design as described in Section 3.2.2. More specifically, we measured the number of retransmissions needed to forward a packet by 1 hop WiFi broadcast, the response time at application level, and the effectiveness of caching. Fig. 3.4(a) shows the CDF for the number of retransmissions for the Info-Traffic application in all the 3 types of mobility aforementioned. For the static case, about 75% of the packets need no more than one retransmission. In the mobility scenario this number goes down to



(a) V-NDN Info-Traffic application CDF number of Link Adaptation Layer retransmission

(b) V-NDN Info-Traffic application: CDF Response time - from the first Interest to the corresponding data

FIGURE 3.4: RTT Analysis

about 65%, however the type of mobility (either on the P8 roof or on the roads) has a negligible impact on the number of retransmissions. 95% of the packets are acknowledged within 5 retransmissions or less (the max-retransmission was set to 7), which is also confirmed by simulation results in Fig.3.6(b). Also note that 15% of packets in the static case were acknowledged before being actually transmitted (therefore they never went on the air), because neighboring cars (presumably in better positions) already forwarded packet (see Section 3.2.2).

The response time CDF for the Info-Traffic application is shown in Fig.3.4(b). In the static case, about 75% of the Info-Traffic Interests retrieved data back in less than 1 second. For the mobility case, experiments with looping around on the roof of P8 performed worse than the experiments on the roads around campus. We believe this is due to the effect of traffic lights, which lumped multiple cars together with relatively long breaks in mobility, thus facilitating data propagation between cars. In addition, P8 is in a midst of WiFi access points and our measurement detected a high noise background.

	Static (#Times)	Around P8 (#Times)	Around Campus (#Times)
Use of Cache	96	3054	1840
Interest Forward	715	4685	13195

TABLE 3.1: Cache vs Forwarding among Consumers and Mules

To better understand the impact of in-network cache, we excluded the data producers from the dataset and analyzed the cache/forward statistics for consumer and mules. Results are shown in Table 3.1. The data shows that caching is more effective during mobility and particularly when mobility happens in a relatively restricted area (rooftop of P8). Observing mules only (i.e. no Interests nor data packets are generated by these nodes), the benefit of caching becomes even more evident: in this case about 66% of the Interests found the

requested data from local cache, thanks to the aggressive caching strategy we proposed in Section 3.1.

We also experimented with the Road-Photo application by letting one of the cars issue an Interest for a photo of a given area. The Interest carries a name "/picture/westwood-strathmore", requesting a picture from a camera that is on board of any vehicles close to Westwood-Strathmore intersection. A vehicle in that area, if any, will turn on the camera, take a snapshot, and send it back. While our design is plain and simple, we were able to retrieve 51 camera shoots. Photos had an average size of 6.3KB resulting on average 5 data packets. A photo is received when all the data chunks arrive at the consumer. The Road-Photo application experienced an average response time of 81 seconds for the mobile case (Fig. 3.3), and 28 seconds in the static case; the median point was 55.6 seconds and 1 second, respectively. We believe that the sparse nature of the cars in the mobile tests affected the response time, as the multi-packet delivery suffered connectivity disruption much more than the Info-Traffic application whose response is made of a single data packet.

3.3.2.2 Robust Data Availability

Once a piece of data has been spread in the network, its availability becomes independent from the connectivity to its producer. Indeed the decoupling of data from its container (its producer) and data caching enable every node that got the data to use it and pass it around to any other consumers who issues an Interest for the data. The following experiment shows a proof of concept: a consumer asks for a content that can only be produced by one car. As soon as the consumer receives the content, the producer is switched off. After that, another consumer issues the same Interest. As expected, the second consumer is able to get the desired content even after the original producer is gone. Furthermore, the response time does not seem to be negatively affected by the absence of the producer thanks to the broadcast nature of the wireless communication that easily allows the spreading of all contents.

3.3.3 V2X Scenarios and the role of the Infrastructure

Because one fetches data by name, V-NDN possesses the innate ability of utilizing node multihoming to communicate with other cars via Ad Hoc WiFi and with servers on the Internet via 3G/4G/WiFi connectivity simultaneously. The current V-NDN implementation simply forwards an Interest through all the interfaces that are available at the time. To observe V-NDN operations in a

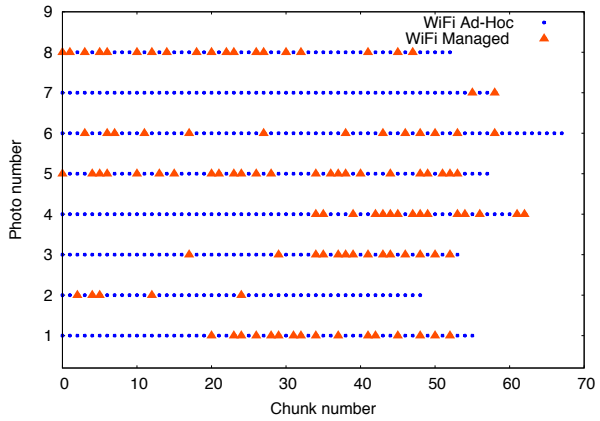


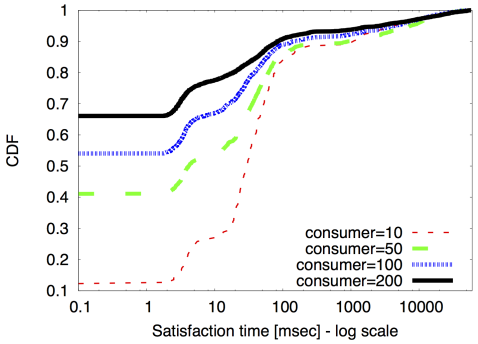
FIGURE 3.5: V2X experiment: communication channels used in receiving photo chunks

multihomed scenario, we performed an experiment of having two cars, one consumer and one producer, running around Parking Lot 32 (P32) in a clockwise fashion. At one corner of P32 we set up a WiFi access point which is connected to the campus network. The *consumer* was equipped with 2 WiFi interfaces, one operating in Ad Hoc mode and the other operating in Infrastructure mode. The *producer* was equipped with 1 WiFi interface configured in Ad Hoc mode and 1 WiMax interface connected to the campus network. We ran Road-Photo application: the consumer requests a photo to be taken by the producer. Interest and data packets were transmitted via all available interfaces. Photos were taken in realtime upon receiving an Interest, their sizes are between 68KB and 100KB. Each photo was split into several data packets of 1300 bytes each. Fig. 3.5 shows on which interfaces the consumer got the contents. The consumer was able to seamlessly receive consecutive chunks of the same pictures from different interfaces via different communication channels.

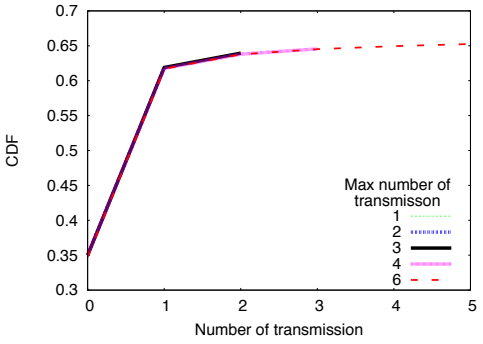
3.4 V-NDN: V2V communication at scale

Since our experiments with real cars are limited in scale, we explored the scalability of V-NDN approach for Ad Hoc communication through simulations. We selected a dense scenario for V2V communication for simulation: an urban area with high vehicular traffic, where cars run in every direction allowed by the roads with different speed, and both traffic jams and empty streets can happen.

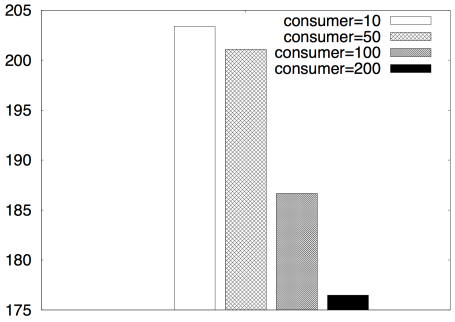
We implemented V-NDN forwarding strategy and LAL in ndnSIM, a ns3-based NDN simulator [7]. The simulation consists of 695 cars moving in a residential area of 2100 meters by 2100 meters in the city of Los Angeles, CA (34.040569,-118.463308). The cars mobility is generated using SUMO [83]. To make the simulated scenario as close to reality as possible, the traffic volume is



(a) V-NDN traffic interest satisfaction time



(b) V-NDN ack distribution per transmission at 2.5 level



(c) V-NDN overhead: number of interest sent in the air for each interest issued by an application

FIGURE 3.6: V-NDN Simulation Results

shaped according to the importance and size of each street. The radio propagation is modeled using CORNER [55], an high fidelity propagation model for urban scenarios that accounts for the presence of buildings as well as fast fading effects. We ran 300 seconds simulated time. All the cars are equipped with a WiFi Ad Hoc interface (802.11a), but only in a subset of them runs the **Info-Traffic** application as either consumers or producers; the others play the role of packet forwarders and/or data mules. We configured 14% of the cars as producers, and changed the number of consumer cars.

3.4.1 Results

Since the V-NDN forwarding strategy design is part of our next work, the simulation uses the rudimentary scheme of greedy packet dissemination along all direction as we mentioned earlier. Thus the current simulation results can only serve as a quick examination to understand V-NDN's feasibility at scale.

Fig. 3.6(c) shows how many Interest are sent on the network every time the application consumer issues an Interest. Fig. 3.6(a) shows how long it takes for a

consumer to get the desired content. Both graphs reveal that, when the number of cars interested in the same information increases, the performance of the entire system improves substantially as measured by satisfaction time and overhead. Indeed by letting all cars cache overheard information, they become data mules. Not only a car can help forward a content that it has just received, but it can also carry the content and answer Interests for the same content in the future. As we described in Section 3.2.2, the LAL performs retransmissions when a packet is not ACKed. Fig. 3.6(b) shows the percentage of packets considered ACKed (the car has heard an implicit ACK from every possible direction - road stems) as a function of the maximum allowed transmissions for an Interest before giving up. 35% of Interests are ACKed before being transmitted, because the cars already have cached the requested data earlier when they overheard it from other cars. Moreover, after the second transmission, further transmissions do not seem to bring much improvement - after the second transmission, if a packet is still not ACKed, it is unlikely to be ACKed with further retries. We speculate that the reason for this could be due to the lack of connectivity at the time.

3.5 Discussion

While cellular networks have been viewed as the only global wireless infrastructure, in reality they suffer from spectrum scarcity and coverage limitations. At the same time vehicles are being equipped with computation power, storage, and multiple communication interfaces via various communication technologies that can be exploited to take advantage of opportunistic connectivity with other vehicles and physically transporting data over distance.

V-NDN, by naming data rather than hosts and decoupling data from IP addresses, can bring substantial benefits to vehicular communication: it removes the isolation between applications and network transport, allowing forwarding nodes to handle data based on application needs. The communication can start spontaneously — the infrastructure for the IP addresses assignment is no longer required. Naming data also enable mules to cache and reuse the content and makes V-NDN resilient to connectivity disruptions that characterize vehicular networks: even when the communication between consumer and producer is interrupted, mules can bring the required data to the consumer over time. Furthermore locally produced data and data with local meaning, as traffic information, no longer need to be transferred to remote servers before being available to neighbor nodes; data that is produced and consumed in loco can remain in loco and be delivered to the consumers along shortest physical paths.

Remaining Challenges

V-NDN is only the first step in exploring NDN for VANET on the field, it shows the benefits of NDN in vehicular networks but it is far from making the "connected car" concept a reality. A number of challenges highlighted by V-NDN experimentations remains to be addressed. For instance, focusing on the V2V communication, a smarter way to forward Interest without flooding the network in all the directions needs to be implemented to make V-NDN scalable and more efficient. Another challenge is data naming. [163] shows that encoding geolocation into names can help direct Interest forwarding for applications using location-based data, however other types of applications, e.g. fetching today's news, are unable to make use of geolocation. Furthermore, more work is needed to address the security and privacy considerations. Finally, in the case of multi-homing, a more detailed study of possible forwarding strategy may be needed to make the best use of multiple network interfaces.

Chapter 4

Navigo: Interest Forwarding by Geolocations in V-NDN

4.1 Introduction

As discussed in Section 3.5, although V-NDN proves the feasibility and benefits of adopting the NDN paradigm for vehicular communication, it highlights some challenges that still need to be addressed to make NDN for VANET a reality. In particular, V-NDN lacks a way to make smart forwarding decisions, but blindly floods the Interest looking for the Content. Such approach is not sustainable in a real deployment. Thus, we design Navigo [60] to steer Interest forwarding towards the data in order to build a viable V2V network and, whenever possible, exploit application's knowledge in localizing the data.

Contributions First, we developed effective solutions to the problem of mapping data names to data locations and forwarding Interest packets along the best path. Second, we designed an adaptive discovery and selection mechanism that can identify the available data sources across multiple geographic areas and can quickly react to sudden changes in vehicle networks. Third, our solution demonstrates the power of NDN architecture applied to vehicular networking and its ability to cope with Ad Hoc mobility and frequent network connectivity disruptions.

4.2 Navigo Design Overview

A fundamental challenge in Navigo design is how to steer Interest towards where data resides. As previously discussed, the highly dynamic connectivity in VANET renders running a routing protocol infeasible. For traffic applications that intrinsically contain location information in data names, [163] demonstrated that one can simply forward Interests toward the geographic location stated in the names, without the need for a routing protocol. Indeed a broad class of automotive applications is intrinsically location-dependent, i.e. the data produced and consumed by them is tied to specific locations. Examples of such applications range from obtaining road traffic updates on a given street to the search for an available parking space. However, [163] requires the forwarding strategy in each node to understand the semantic of the names to be able to extract the destination information. Such assumption ties the name design with a specific convention, and most important, is not feasible in the current NDN framework, where the forwarding strategy is unaware of the name semantic. A mean of letting consumers suggest where the data may reside is still missing. Furthermore, other types of applications, such as music sharing or data fetching in general, are not associated with any specific locations.

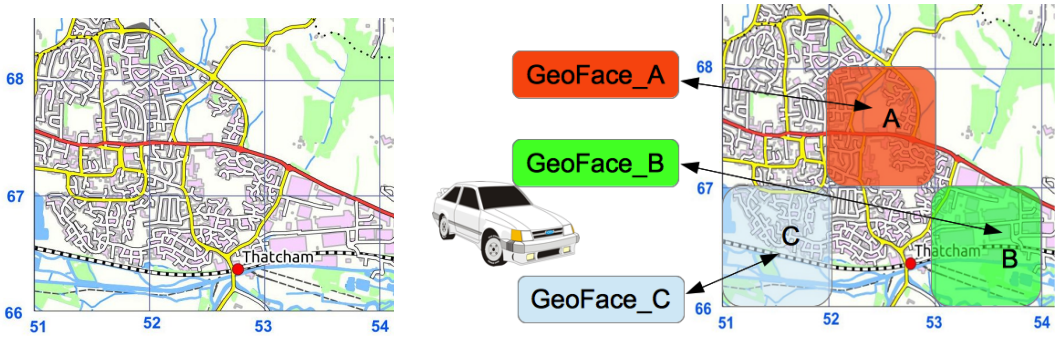
To effectively forward Interests for all types of applications without a routing protocol, our solution is to couple their data names with the locations of where the data resides. While for the first type of application we can bound names with the location the consumer is interested in, for the second type our solution is to bind them with the location of the data provider: either the content producer or a vehicle which is carrying the data in its cache (mule) or an Internet access point (i.e. RSU). This will allow us to do geo forwarding to support all types of applications.

There are a number of specific issues to address to make the above idea work. First, one must define a namespace for geo locations; this is addressed below. Second, one must provide effective means to map data names to locations, which is discussed in Section 4.2.2. Third, we would like to support geo forwarding without modification to the existing NDN forwarding framework; this is addressed in Section 4.2.3.

4.2.1 Naming geographic areas

We divide the world into regions according to the Military Grid Reference System (MGRS). This system is derived from the Universal Transverse Mercator

(UTM) and from the Universal Polar Stereographic (UPS) grid systems, where each region is identified by a label (Figure 4.1(a)).



(a) Example of MGRS map

(b) Mapping GeoFaces to geographic areas

FIGURE 4.1: Navigo: geo-areas and GeoFaces

In the current implementation geo-areas have a fixed size (200x200 meters) but the MGRS scheme easily allows names with different precision levels, by adding or removing digits from the name (or building areas with multiple "MGRS blocks"), e.g. 4QFJ 12 67 defines a 1Km precision, while 4QFJ 123 678 has a 100 meter precision. Further analysis regarding more flexible areas is left for future research.

4.2.2 Mapping data names to geo-areas

When a node has an Interest in hand for data without any geographic meaning and has no knowledge about the prefix of the data name (not in its FIB), the node simply sends the Interest out in all directions. If any of these flooded Interests hits a copy of the matching data, the responder attaches its geo-area MGRS name (e.g. 4QFJ 123 678) to the returned Data packet. As the Data packet follows the breadcrumb trace of the Interest, all the nodes along the way learn the binding between the data name prefix and the corresponding geo-area. This information allows them to forward future Interests for the same name prefix towards geo-areas bound with that prefix only. For location-dependent data, the consumer can avoid the initial Interest flooding procedure by binding the Data name with the geo-area the consumer is interested in before sending the Interest.

4.2.3 Hiding geographic forwarding from basic NDN framework

The current NDN forwarding daemon has no concept about geo-areas. Indeed its FIB contains ⟨prefix, face⟩ pairs only. To exploit the binding among names and

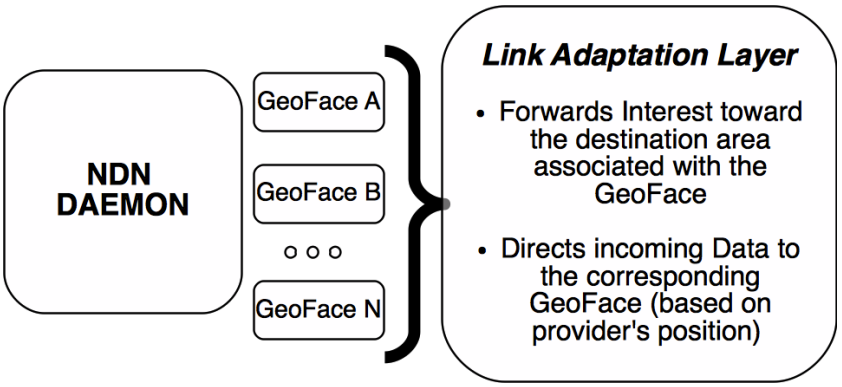


FIGURE 4.2: NDN and GeoFaces

geo-areas while using the current NDN architecture Navigo introduces the concept of "Geographic Faces" (GeoFaces), and implements the binding of name prefixes to geo-areas through a two-step process. It first binds a geo-area to a GeoFace, and then lets the FIB store the mapping from the name prefix to the GeoFace as in the current NDN architecture. The same prefix can be mapped to different faces, enabling Navigo to exploit multiple data sources whenever the same content can be retrieved from different areas, by binding multiple GeoFaces to the same prefix in the FIB (section 4.3 describes the geoFace selection process).

One (i.e. the consumer) can now simply register a new rule in the FIB to bind a prefix with a specific geo-area.

An extended version of the LAL, introduced by [59], stores the GeoFace to geo-area mapping. Such GeoFaces are an abstraction of the WiFi ad hoc interfaces the car is equipped with. Navigo extends the LAL by implementing a forwarding mechanism which, given a GeoFace F_X bound to a geo-area X, steers the Interest over the V2V channel along the best path to X, according to Section 4.5. Furthermore, the LAL sorts incoming Data packets and redirects them to the correct GeoFace based on the Data provider's location.

4.2.4 Design assumptions

This chapter is focused on urban scenarios. We assume all vehicles are equipped with a GPS sensor and a digital map, therefore able to identify their own location (and thus its MGRS name). Each vehicle is also equipped with a WiFi network interface (set up in Ad Hoc mode) and with enough storage and computational capabilities; we leave the study of the performance under constrained caching to a future work. Furthermore, we assume the presence of

RSUs along some roads which provide Internet connectivity to all the cars and which are running an NDN stack and Navigo.

4.3 GeoLocation-based Interest Forwarding

The main idea behind the forwarding strategy adopted by Navigo is to explore the area surrounding the node looking for producer, mules or RSUs and then, as soon as the first Data packet comes back, forwarding future Interests for the same prefix towards the geo-area where the data is coming from.

Forwarding algorithm When forwarding an Interest, Navigo tries to exploit the presence of multiple data providers to balance traffic and make communication more resilient to the high dynamism of VANET, which limits the validity in time of a single rule. To forward an Interest I for a Content named N Navigo adopts the following forwarding strategy:

- If I does not match any entries in the FIB (there is no information about the Data location), the Interest is sent using a flooding technique (see 4.5) without specifying any destination area—the node is in a so-called *exploration phase*.
- Instead, if several GeoFaces are bound to N, the forwarding strategy selects one face in a round-robin way.
- If only one face is available, with probability p (0.95 in our experiments) the GeoFace in the FIB is used, while with probability $1 - p$ the node acts as in the *exploration phase*, flooding the Interest. Navigo adopts this additional exploration phase to avoid focusing on a single destination area for a long time while some other opportunities may come with mobility to balance the traffic.
- After sending an Interest, if the node does not receive the requested data before a deadline T (300ms in our experiments), the binding in the FIB among the face and N is removed. If N is bound with this face only, the node is back in the *exploration phase* for N.
- If multiple Interests with the same prefix N are sent in pipeline on the same GeoFace F_X, whenever one of them is satisfied, the deadline is removed for all the pending Interest for N: receiving a Data packet indeed means the information is still available in that geo-area, although some of the Interests may fail.

An evaluation of using more sophisticated criteria, such as round trip time or success rate, to select the best face is left for future works.

4.4 FIB Management

4.4.1 Binding content location to the right name

Names in NDN are hierarchical: a Content named */N* may be divided in several chunks i.e. */N/c1*; */N/c2*. To correctly forward Interest for all these pieces (and only them), one must have a FIB entry with the prefix */N* (FIB lookup is done by longest prefix match). Consumer and producer know the name semantic and thus can identify which prefix aggregates all the pieces of a Content. But forwarders, which may not be running the application, are unaware of the semantic and thus cannot correctly fill the FIB[1]. To address this issue, the consumer's LAL attaches to each Interest (into a 2.5 layer header defined by LAL) the prefix which aggregates all the pieces of the requested content. This information is then spread by the forwarders, allowing every LAL to register the correct rule in the FIB whenever the Interest is satisfied.

4.4.2 FIB size

A FIB entry stores prefix and a list of faces that can be used to retrieve such Content. Associating geographic areas with faces, the geo-area dimension may affect the size of the FIB: smaller areas means higher probability to receives Data from different regions, especially when the data provider is a moving car, which increases the number of faces bound with the same prefix. At the same time, however, if the region is too large, the FIB is smaller but the portion of area where cars flood the Interest increases, leading to a larger overhead.

With 200×200 meter areas our experiments show that the number of faces bound with the same prefix never reaches values higher than 9. Indeed, removing the binding among a GeoFace and a prefix as soon as an Interest fails stops the node from having too many faces bound to the same prefix.

[1] Assuming that by removing the last component of the name one gets the correct prefix to aggregate all the Interests for the Data may not always be true.

4.5 Link Adaptation Layer

Navigo extends the original version of the LAL presented in [59] exploiting the knowledge of the destination area. It takes care of the GeoFaces to geo-areas binding, it steers Interest along the shortest path to the destination region and it copes with urban scenario characteristics by taking into account the presence of obstructions in contrast to more suitable places for wireless propagation.

4.5.1 LAL and GeoFaces

The LAL creates and destroys GeoFaces and keeps the mapping between GeoFaces and geo-areas in the Face-to-Area table (F2A). Whenever a node receives a Data packet, the LAL extracts the geo-areas information attached by the data provider. If this area is not associated with any faces, the LAL creates a new GeoFace and binds it to the geo-area (adding the relation to the F2A). Whenever a GeoFace is unused for a certain amount of time (i.e. order of tens of seconds), the LAL removes the face and any references to it from the F2A and the FIB.

4.5.2 Calculating the shortest path

The shortest path to the destination area is calculated applying a specialized Dijkstra algorithm with the street topology as underneath graph: streets are edges and intersections are nodes of the graph. Conceptually Navigo deliberately substitutes the more stable road-topology to the network topology which in VANETs is dynamically partitioned and features short lived links lasting only few seconds on average [129]. In computing the shortest path, LAL is unaware of neighbors locations. Indeed, to avoid the overhead of periodic exchanges of 1-hop neighbor position[56], Navigo doesn't use any neighboring protocol. It relies on a probabilistic approach to minimize the chances of hitting an empty road. Navigo assigns costs to edges that are inversely proportional to the number of lanes[2]. In this way the algorithm tends to prefers paths with larger roads and more likely to have running cars at any moment thus leading to a more stable path. Furthermore, the algorithm takes into account the presence of obstructions of wireless communication and merges roads that are in line of sight while splits paths that require a turn. Indeed turns mean additional hops in the transmission, which increases overhead and delay.

[2] After preliminary simulation, we adopted the following costs: 1 for 2-lanes road, 0.7 for 4-lane street and 0.25 for 6-lanes roads. Analysis of other factors, such as amount of data traffic or cars to determinate the weights of a road are left for future works.

4.5.3 Forwarding process

Whenever the NDN daemon sends an Interest through a GeoFace, the face passes the packet to the LAL, which performs a lookup on the F2A to determine the destination area name. This information is then encoded within the L2.5 header that encapsulates the Interest, together with the position of the node, spreading the information to all the neighbors in the transmission range. Once a car receives an Interest, the LAL extracts and stores the information about the destination area, the position of the previous node and the nonce of the Interest and then it passes the packet to the forwarding strategy (see algorithm 1). If the NDN daemon decides to forward the Interest on the V2V network, it passes the packet back to the LAL, either by using the GeoFace specified in the FIB[3] or by using the exploration phase procedure (see the following paragraph on Flooding). The LAL, based on the nonce of the Interest, recovers the position of the previous hop and the destination area specified by the original consumer, which is used as Interest destination. LAL overrides any local decision about the destination area with the consumer's will. The analysis of benefits and challenges of overriding the consumer's will with local information is left for future works.

Given the destination area, LAL computes the shortest path algorithm and forwards the Interest only if it's closer (path is cheaper) to the destination area than the previous hop. Once the Interest reaches the destination area Navigo uses the protocol described in [59] to perform a local Interest dissemination, flooding the Interest in all the available directions. Cars outside the destination area may reply with the Data, but they don't forward the Interest anymore, constraining its dissemination to the destination area only. Whenever the Interest hits a data provider, the node replies with the Data, attaching its location MGRS name X to the 2.5 layer header. As defined by the NDN protocol, Data will follow the breadcrumbs left by the Interest in the PIT of every nodes it passed through. In this way back to the consumer, LAL updates the FIB, binding the Content prefix with the GeoFace associated with X and forwards the Data only if it is closer than the previous hop to the node from which it received the Interest.

To increase the communication reliability, during the packet forwarding process LAL utilizes the implicit acknowledgment concept as in [59], requiring an implicit ACK from the street indicated by the shortest path as next hop.

[3] If necessary, Navigo allows the outgoing and incoming faces to be the same.

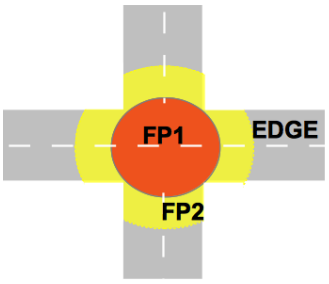


FIGURE 4.3: Intersection as forwarding points.

Flooding

When the forwarding strategy is in *exploration phase*, it selects the v2vFace as outgoing face for the Interest. Such v2vFace, introduced by [59], is used to spread the Interest over the V2V channel in all the directions. The destination area is not specified and the LAL adopts the packet suppression techniques defined in [59] to flood the network.

4.5.4 Forwarding based on forwarding points

As adopted in [52], due to the presence of buildings that obstruct wireless communication, the best strategy to cover a large area with the smallest number of hops is selecting forwarders at the intersections. Navigo capitalizes this observation and uses the junctions as preferred forwarding points, by speeding-up the transmission of cars within an intersection. As [52], Navigo splits every junction into two parts: the core, so-called FP1, and the external area, so-called FP2 (see figure 4.3). Among vehicles inside the same junction, LAL privileges cars within FP1, which have a more central position, increasing the chances to reach more cars in one shot.
When forwarding a packet, to maximize the progress at each hop the LAL in [59] assigned higher priority to nodes far from the sender by speeding up their transmissions. Navigo extends this approach, by taking into account not only the distance by the previous hop, but also the presence of forwarding points and the type of packet:

- Vehicles inside a Forwarding Point (FP1 or FP2) wait less. A random component is added to the waiting timer to avoid collisions among packets transmitted by cars at the same junction. Furthermore, cars inside FP1 wait less than vehicles in FP2.
- Among cars located in different Forwarding Points, the shorter the distance to the previous hop, the longer the wait: the road among the current and

Algorithm 1: LAL – Interest Forwarding

```
/* LAL receives an Interest                                                    */
Data: Interest I.
Nonce ⟵ ExtractNonce(I);
Extract PHPos (previous hop position) from 2.5 header;
Extract DA (destination area) from 2.5 header;
GeoFace ⟵ Lookup(F2A, DA);
/* Create GeoFace for DA if needed                                             */
Store Nonce, PHPos, DA in InterestFromNetwork;
Pass I to forwarding strategy using GeoFace;
/* If DA is not specified by the consumer (exploration phase), use v2vFace     */
**********************************************
/* LAL receives an Interest from the forwarding strategy                       */
Data: Interest I; my position MyPos; face used F.
Nonce ⟵ ExtractNonce(I);
if Nonce ∈ InterestFromNetwork then
    Distance ⟵ CalculateDistance(MyPos, PHPos);
    if DA defined for I then
        /* Calculate path cost                                                 */
        PrevHopCost ⟵ Dijkstra(PHPos, DA);
        ⟨NextHop, Cost⟩ ⟵ Dijkstra(MyPos, DA);
        if Cost < PrevHopCost then
            CalculateWaitingTimer(Distance, MyPos);
            AttachToPacket(I, MyPos, DA);
            Send(I);
            WaitForAckFrom(NextHop);
        else
            Stop processing I;
        end
    else
        /* No DA specified in I                                                */
        CalculateWaitingTimer(Distance, MyPos);
        Send(I);
        WaitForAckFrom(AllPossibleDirections);
    end
else
    /* I generated locally                                                     */
    if F ∈ GeoFaces then
        DA ⟵ GetCoordinates(F);
        ⟨NextHop, _⟩ ⟵ Dijkstra(MyPos, DA);
        CalculateWaitingTimer(Distance, MyPos); AttachToPacket(I, MyPos, DA);
        Send(I);
        WaitForAckFrom(NextHop);
    else
        /* Exploration phase on v2vFace                                        */
        Send(I);
        WaitForAckFrom(AllPossibleDirections);
    end
end
```

the previous hop is divided in 100 meters sections. Cars in the Forwarding Point within the furthest sector from the previous hop (distance greater than 500 meters) wait for the minimum waiting timer. Getting closer to the previous hop, each 100 meters segment adds a constant value (4 ms for Data, 1.5 ms for Interest) to the waiting timer.

- The waiting timer for cars at an edge is inversely proportional to the distance to the previous hop (as in V-NDN).
- The maximum waiting timer for a Data packet is smaller than the minimum waiting timer for an Interest. Speeding up the transmission of Data stops the data provider's neighbors to propagate the Interest any further.

The entire process of calculating the waiting timer is self-deterministic: each car calculates its own waiting timer based on its position and the distance to the previous hop, without requiring any knowledge about neighbors position.

Figure 4.4 shows how much a car waits before forwarding a packet[4]. It must be noted that even though it might take at most 50 ms to make a one-hop progress for a packet, the delays sensitively reduce when a Data packet has to be forwarded or when the car trying to forward an Interest is either at an intersection or far from the previous hop. Furthermore, based on the implicit acknowledgement policy already discussed, as soon as one car forwards a packet, all the vehicles with larger waiting timers suppress their transmission.

4.6 Simulation

4.6.1 Scenario

For the initial evaluation of our design we considered an urban road network with both residential streets and major arteries. The map we chose for the simulations spans a 2.1 × 2.1 km area in the city of Los Angeles, CA. The vehicular micro-mobility traces were generated with SUMO [83]. In order to make the simulated scenario as close to reality as possible, the traffic volume is shaped according to the importance and number of lanes of each street: out of 812 cars, 48% of them were on 6-lane roads, 37% on 4-lane roads, and the remaining 15% was on 2-lane residential streets. The average time a vehicle spends inside the simulated area is 3 minutes.

[4]The values shown in Figure 4.4 correspond to the values used in the experiments. While for simplicity these values are constant, adapting the waiting timer based on the environment (e.g. car density, data traffic, ...) may improve the performance.

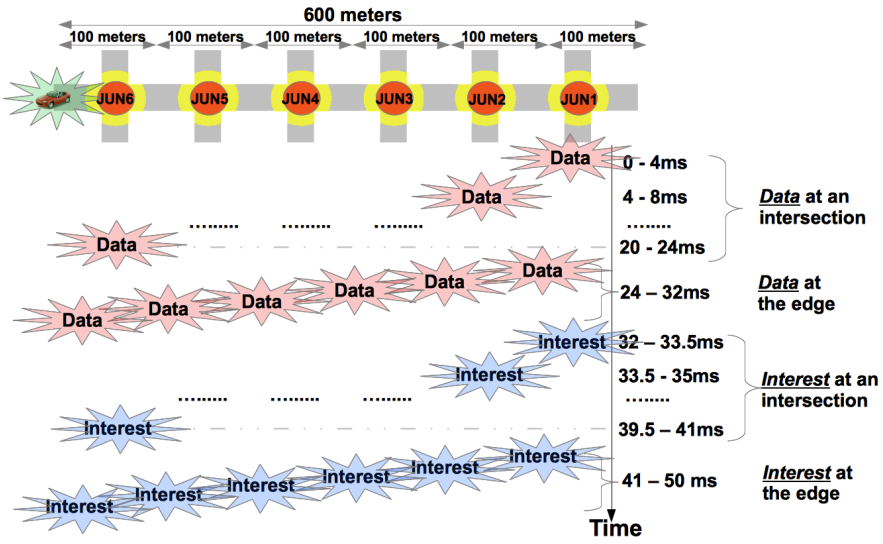


FIGURE 4.4: Navigo speeds up Data transmission and prioritizes packet forwarding at the intersections. Cars close to the previous hop and cars not located within a forwarding point that are waiting to send the same packet will suppress their transmission.

For simplicity, cars enter the map with an empty Content Store, even though this configuration penalizes Navigo, which heavily relies on nodes' caches. Moreover, similarly to [59], we observed that deploying large storage devices in cars should not be a problem nowadays, and therefore we set the Content Store size limit to 10 GB on each car, which means the car can keep the received data in the cache for the entire duration of its trip.

To evaluate the effectiveness of the content discovery process, cars enter the map with an empty FIB.

All vehicles are equipped with an IEEE 802.11a wireless network interface operating at 24 Mbit/s, configured in ad hoc (IBSS) mode on the same fixed channel used by the roadside units (RSUs), thus allowing cars to communicate with both other vehicles and RSUs via the same interface.

The roadside units are positioned to mimic the location of the actual access points deployed by Time Warner Cable in the same area. We selected only a subset of 4 out of the 21 access points currently deployed in that area [148]. In our model RSUs are assumed to be fully functional NDN nodes; furthermore, we assume that no MAC-layer authentication or link setup process is needed. Although this does not reflect current WiFi practices, emerging standards such as the vehicular-specific IEEE 802.11p and the proposed IEEE 802.11ai either do not require link-layer connection establishment, or they reduce the link setup time to less than 100 milliseconds.

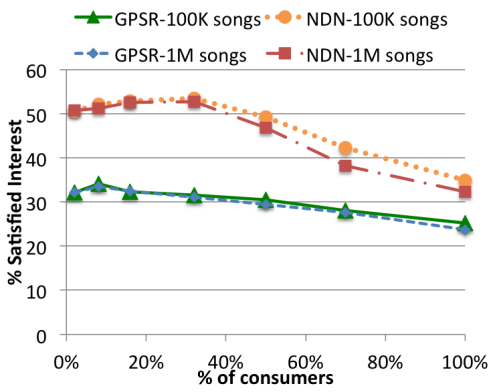
We implemented Navigo in ndnSIM [7], starting from the implementation of V-NDN (for ndnSIM) described in Chapter 3. Navigo implementation extends the V-NDN LAL and forwarding strategy with the additional elements described in this chapter, adding a new type of face — the GeoFace —, extending the LAL with new timer and level-2 forwarding policies, and implementing a new forwarding strategy. The radio signal propagation was modeled with CORNER [55], a high-fidelity propagation model for urban scenarios that accounts for the presence of buildings as well as fast-fading effects.

4.6.2 Music streaming over NDN

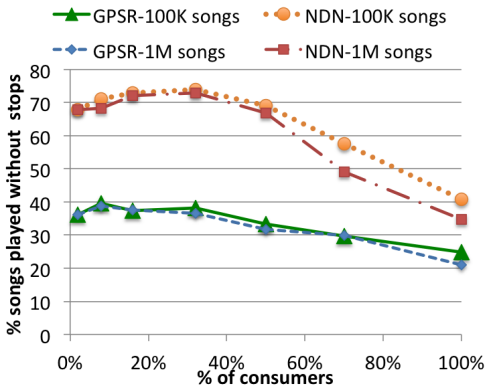
In order to evaluate Navigo performances, we devised a "music streaming" application: a hypothetical Internet music streaming provider that can be reached by any of the 4 RSUs deployed on the map via a 100 Mbit/s wired channel. The client application (consumer) runs on a subset of all the cars; we varied the cardinality of the subset from 2% to 100% across our simulations.

Each song has an average length of 3 minutes, yielding about 1700 chunks of data per song, if we assume an average encoding bit rate that is common among current commercial music streaming services such as Spotify. Requests for songs are generated according to a Zipf distribution with an α parameter obtained from [84], where Kreitz *et al.* found that the top 12% most popular songs in the library are requested 88% of the time. When a song is chosen, the consumer starts issuing Interest packets progressively for every chunk of which the song is composed. To improve the performance we implemented a simple mechanism for request pipelining, with a hard limit of 20 pending Interests (i.e. expressed but not satisfied) at any given time.

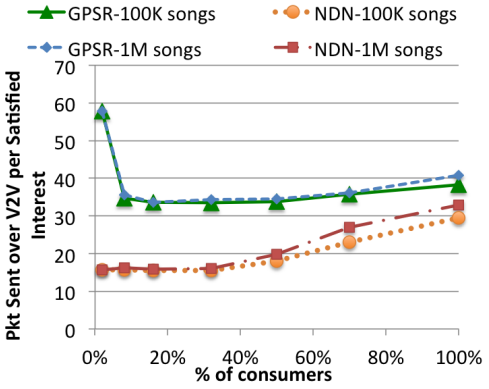
The application tries to provide the best possible user experience, thus its main goal is to successfully retrieve a chunk before the playback reaches that point of the song. In order to do so, and since Data packets can arrive out-of-order, the streaming client maintains a buffer of song fragments that have already been fetched but have not yet been played. When this buffer underflows, the application has to pause the playback and wait for the missing chunk before playback can be resumed. This event is highly undesirable since it leads to a poor user experience. In our simulations we recorded whether an underflow occurred during the playback of a song. We believe this metric provides an important tool to evaluate the success of our solution.



(a) Success rate



(b) User satisfaction



(c) Protocol overhead

FIGURE 4.5: Performance with different music library size

4.6.3 Simulation results

We compared Navigo to GPSR [76], a well-known routing protocol for mobile wireless networks that uses the geographic positions of nodes to make packet forwarding decisions. GPSR typically requires a location service to discover the data source position: in our experiment we provided GPSR with a cost free oracle able to locate the closest node (server or consumer) with the requested chunk of the song. In this section we present the results obtained from the simulations and we analyze them.

4.6.3.1 Success rate

Defined as the ratio between the number of satisfied Interests and the number of Interests issued by all consumers. The results for this metric are shown in Figure 4.5(a). Navigo is able to satisfy a much higher percentage of Interests compared to GPSR, and although the margin of improvement shrinks with 70% and 100% of consumers, Navigo can still satisfy 10% more Interests than GPSR.

4.6.3.2 User satisfaction

As explained in 4.6.2, this metric is expressed as the percentage of songs played without any interruptions caused by a playback buffer underrun. During our simulations the maximum buffer size was set to 30 seconds. Figure 4.5(b) shows that Navigo substantially outperforms GPSR, especially with 50% consumers or less.

4.6.3.3 V2V channel access (protocol overhead)

Represents the intrinsic "cost" of the V2V protocol in terms of number of accesses to the WiFi channel needed to satisfy an Interest, defined as the ratio among the number of packets sent on the V2V channel and the number of Interest satisfied. For Navigo this means the average number of Interest (satisfied and not) and Data packets that were sent on the air for each satisfied Interest. For GPSR we counted also ARP, ICMP, and Hello packets, which are required to run the protocol and therefore part of its overhead. We can see in Figure 4.5(c) that GPSR requires a much higher number of accesses to the V2V channel compared to our solution. In particular, while GPSR always needs to send more than 30 packets for each satisfied Interest, Navigo requires less than 20 packets in most cases, increasing the overall network efficiency, and only becomes slightly worse with more than 50% of consumers.

4.6.3.4 Load on the infrastructure

Expressed as the number of requests received by the streaming server (located on the Internet behind the RSUs) divided by the total number of Interest satisfied. This metric is particularly interesting as it illustrates a major limitation of IP's approach. Indeed, with IP-based protocols, it sometimes happens that the request reaches the content provider but the response fails to travel back to the consumer. In this case the consumer has to re-request the data from the content provider, because there are no caches along the path and, from the point of view of IP, the two requests are completely different and unrelated, even if they refer to the same content. This of course does not happen with NDN: Interests re-issued after a timeout can be satisfied by any other node that cached the desired Data packet during the previous failed attempt(s). The effect is evident from Figure 4.6. The inability of GPSR to exploit in-network caching results in a load ratio around 1.2 or higher in all scenarios. On the other hand Navigo never goes above 1, and in most cases the load is around 0.8. Moreover, as the number

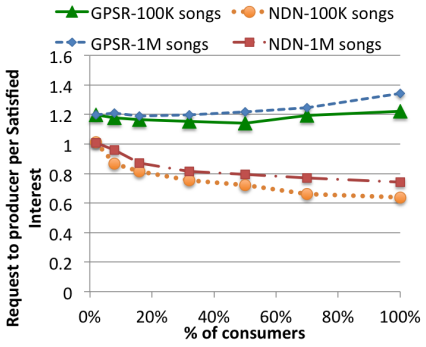


FIGURE 4.6: Load on the infrastructure with different music library size

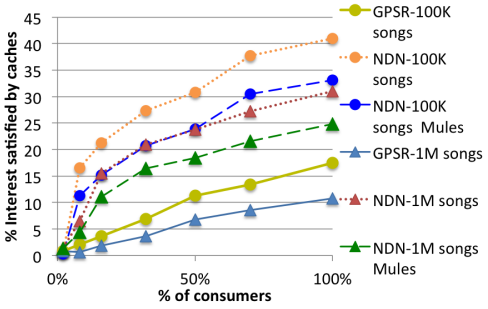
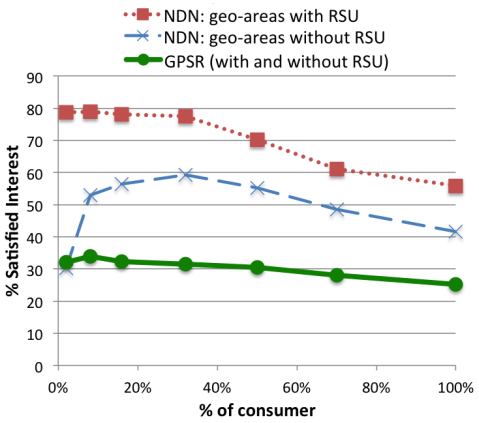


FIGURE 4.7: Infrastructure offload varying music library size

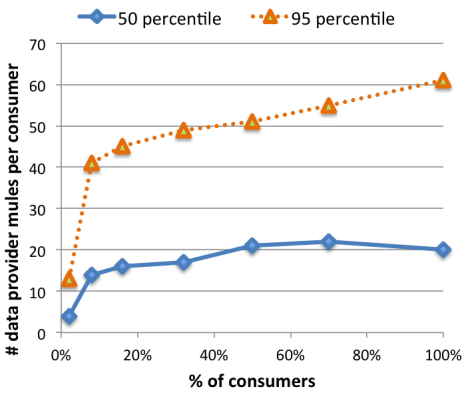
of consumers grows, our solution is able to take advantage of the increased caching opportunities, thus lowering the load on the infrastructure even more, contrary to GPSR where the load slightly increases.

4.6.3.5 Infrastructure offload

Measures the effectiveness of in-network caching for reducing the load on the infrastructure. Concretely, this is defined as the percentage of Interests satisfied by a Data packet coming from the cache of a node (either car or RSU). In the GPSR case only those nodes where the streaming application is running are able to act as caches. The results are reported in Figure 4.7, with an additional pair of lines, labelled "mules only" in the legend, where we considered only other cars as potential caches (i.e. RSU caches were excluded). As expected, Navigo leads to a substantially higher cache utilization, even when only mules are considered, while GPSR does not go beyond 20% of Interests satisfied by caches. Note that to avoid skewing the results to our advantage, the effect of caches on re-issued Interests described for the previous metric is not considered here.



(a) Success Rate sending Interest towards geo-areas with and without RSUs



(b) Consumers opportunistically retrieve a song from different mules

FIGURE 4.8: Navigo simulation results: mobile nodes as data provider.

4.6.4 Handling mobility of data providers

When data is provided by moving mules, the binding between names and geographic areas may have a short life. Navigo partially copes with this problem by specifying an entire geographic area as destination, instead of an exact point in space, and then by flooding the Interest within the destination area. More importantly, as shown in Figure 4.8(b), a consumer is able to opportunistically retrieve chunks of the same song from different moving mules. On one hand, enabling every receiver to cache a Data packet allows the content to spread to a large number of cars; on the other hand, NDN, focusing on content rather than nodes, makes the identity of the data provider irrelevant. These two factors combined allow the consumer to retrieve the data in an opportunistic way from nearby nodes without having to "follow" a specific node.

Moreover, to validate the effectiveness of Navigo approach to the mobility of data providers, figure 4.8(a) shows the Interest success rate when a consumer sends a request towards an area which includes a RSU and when the Interest is sent towards a geo-area without RSUs, thus towards one or more mules. In this evaluation, the success rate of the exploration phase (Interest sent in all the directions) is not taken into account. The success rate when Interests are sent towards areas with RSUs is clearly higher, mainly because (1) the RSU doesn't move while mules do; (2) the RSU is connected to the music producer, thus has access to any content, while mules can answer only for Data packets they store in the CS. Nevertheless, with a small penetration rate of the application as 8%, the success rate of Interests sent towards geo-areas with only mules goes from 40% to 60% and, most important, is higher than the success rate of GPSR (which includes both Interest sent towards RSU and towards other consumers).

Traffic Application

To analyze how Navigo takes advantage of the location information when they are intrinsically attached to contents, we implemented a "traffic-update" application: producers (cars) autonomously collect information about the traffic on the road they are traveling on (i.e. vehicle speed), while consumers (cars) request traffic status updates specifying in the Interest name the location (i.e. a portion of a road) they are interested in. Such data has an intrinsic geographical meaning, thus the exploration phase can be avoided by letting the consumer bind in the FIB the content prefix with the geo-area including the location of interest. Table 4.1 shows the success rate of the traffic application when 10% of the cars running on the map are producers. The success rate is defined as the ratio among the number of Interest satisfied and the number of satisfiable Interest. An Interest is considered as "satisfiable" whenever, at the moment the consumer issues the request, there is at least one producer somewhere in the map (it doesn't matter where) that can satisfy the request (Interest for data that has not been produced or for data that has been produced by cars which have already left the map are not considered in this statistic).

% of consumer	1%	10%	30%	50%
Succes Rate	19%	36%	54%	63%

TABLE 4.1: Traffic Application: Success Rate with 10% of producers

While this is just a toy scenario and the way the data is produced is far from the reality, these results indicate that, even with a small penetration rate of the application (10% of producers), a reasonable success rate (over 50%) is reachable without any support from the infrastructure and without any exploration phase. Whenever data is tied with the location where it is generated and can be produced by multiple nodes, steering the requests towards the location of interest can be enough to reach a producer or a mule and retrieve the content.

4.6.5 Simulations with higher car density

We performed the same set of simulations for the music application described in 4.6.3 using a denser car mobility. This time the total number of cars on the map was 1048, arranged as follows: 56% on 6-lane roads, 30% on 4-lane roads, and the remaining 14% on 2-lane roads. The rest of the parameters were left untouched.

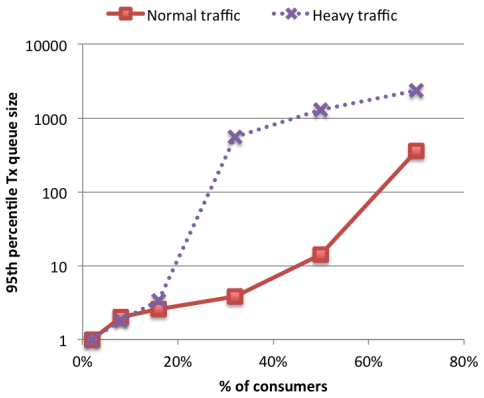


FIGURE 4.9: Transmission queue length with different car density

The results relative to the success rate and user satisfaction, although decreased compared to the previous mobility, clearly showed that Navigo can perform substantially better than GPSR even with a larger number of nodes. The percentage of infrastructure offload raised even more, due to the fact that Navigo can take advantage of the improved caching opportunities offered by the denser car traffic. However, Navigo's overhead, measured in terms of number of V2V channel accesses, also increased, and reached the same level of GPSR in the scenarios with more than 32% of consumers.

We speculated that this performance degradation was to be ascribed to a rapid worsening of the wireless channel conditions: as more and more nodes try to transmit, the network becomes congested, the chance of collisions increases and more packets are dropped due to queues filling up.

To confirm this intuition we measured the length of the transmission queues at the MAC layer on each node. Indeed, as Figure 4.9 shows, starting with 32% of consumers the queue length increases by two orders of magnitude in the heavy vehicular traffic scenario. By comparison, the increase is much slower with the previous mobility. We believe that these findings satisfactorily explain the reduced protocol efficiency observed in the high density simulations. We intend to address this limitation of Navigo in a future work, by investigating congestion avoidance and congestion control techniques.

Moreover it should be noted that, while GPSR packets always follow a single path, Navigo may experience cases of multipath, because forwarding decisions are taken at the receiver side. For instance, cars on different roads might decide to forward the same packet if they cannot hear each other, because both of them are closer to the destination than the previous hop. This event can increase the overhead, but at the same time it makes the protocol more reliable, thus increasing the chances of retrieving the desired content.

4.7 Discussion and final remarks

Previous work V-NDN showed the feasibility of NDN in VANETs and evaluated benefits and challenges of the named-data approach in those highly dynamic environments. However, it failed to address some of the issues typical of the V2V and V2I communication, as, for instance, data discovery, scalable forwarding strategies, etc. which prevents the deployment of V-NDN at large scale. Navigo started from there and addressed some of the limitations of V-NDN, with the goal of improving the efficiency of the communication and make V-NDN more scalable, primary requirements for the realization of the "connected car" concept.

With Navigo we developed a self-learning scheme to enable effective data delivery in highly dynamic vehicular environments. Navigo strategy is to learn where the Content resides and then steer Interests towards such an area. In contrast with IP-based geo-routing, which attempts to deliver packets to a specific end node, Navigo forwards Interests towards the area content resides in, enabling fetching from any available data carrier within the region, either producers, mules, or RSUs. Navigo automatically learns content's geographical location and requires no location service or oracle which are typically required by traditional Geo-routing. Furthermore, while IP-based geo-routing is connectivity-dependent and uses a one-hop hello protocol to maintain the local topology, all Navigo traffic is related to Interest–Data transactions, i.e. if there is no request for content, there would be no packet in the network. Lastly, we observed that the NDN's basic breadcrumbs mechanism is resilient to mobility: the 95^{th} RTT percentile for an Interest-Data transaction is less than 300ms. Vehicles do not move far in the time elapsed between an Interest and the corresponding data thus ensuring the validity of the Interest breadcrumbs in the PIT and the effective retrieval of Data packets.

Navigo has been extensively evaluated through simulations and features low overhead and high performances for both V2V and V2I scenarios. Our simulation setting assumes that all RSUs can listen to the packets within their vicinity of WiFi signal reachability. We understand that the situation can be different in real WiFi deployment today, where RSUs may not be in the same SSID domain as vehicles and thus may not be able to receive/send packets with cars. However we believe this issue is simply the artifact of today's protocol implementation, while our goal in this work is to explore what is achievable by NDN based V2V, without the constraints of today's implementation.

Chapter 5

Producer Mobility: MAP–ME

5.1 Introduction

With the phenomenal spread of portable user devices, mobility has become a basic requirement for almost any communication network (not only VANET) as well as a compelling feature to integrate in the next generation networks (5G). The need for a mobility-management paradigm to apply within IP networks has striven a lot of efforts in research and standardization bodies (IETF, 3GPP among others), all resulting in a complex access-dependent set of mechanisms implemented via a dedicated control infrastructure. The complexity and lack of flexibility of such approaches (e.g. Mobile IP) calls for a radically new solution dismantling traditional assumptions like tunneling and anchoring of all mobile communications into the network core. We thus extends our work on NDN and VANETs to address network mobility for more generic nodes, as smatphones, tablets etc.

As already demonstrated before, the Named Data Networking paradigm brings native support for mobility, security, and storage within the network architecture, hence emerging as a promising 5G technology candidate. Specifically on mobility management, NDN has the potential to relieve limitations of the existing approaches by leveraging its primary feature, the redefinition of packet forwarding based on *names* rather than on *network addresses*. We believe that removing the dependence on location identifiers is a first step in the direction of removing the need for any anchoring of communications into fixed network nodes, which may considerably simplify and improve mobility management.

As a direct result of NDN design principles, consumer mobility is natively supported: a change in physical location for the consumer does not translate into a change in the data plane like for IP. The retransmissions of requests for not yet

received data by the consumers take place without involving any signaling to the network. Producer mobility and realtime group communications present more challenges, depending on the frequency of movements, latency requirements, and content lifetime. The topology does not reflect the naming structure, and we have to preserve key functionalities such as multipath, caching, etc. In all cases, beyond providing connectivity guarantees, additional transport-level mechanisms might be required to protect the flow performance, which are beyond the scope of this work (see [39] for instance).

Tackling such problems, in a simple and effective way by exploiting NDN key characteristics is at the core of this chapter. Previous attempts to address global and local mobility have been made in NDN (and ICN in general) literature to go beyond the traditional IP approaches, by using the existing NDN request/data packet structures to trace producer movements and to dynamically build a reverse-forwarding path (see [180] for a survey). While most of them support not only local, but also global producer mobility, they still rely on a stable home address to inform about producer movements (e.g. [181]) or on buffering of incoming requests at the producer's previous point of attachment – PoA – (e.g. [77]), which prevents support for latency-sensitive streaming applications. We look at local mobility and focus on this class of applications (e.g. live streaming or videoconferencing) as they have the most stringent performance requirements: negligible per-packet loss-rate and delays. In addition, they typically originate from a single producer and don't allow for the use of caching. Good performance for other classes such as adaptive or elastic flows is simpler to guarantee as they only require a high-enough average throughput [127, 137]

In this chapter, we aim to take one step forward in the definition of a name-based mechanism operating in the forwarding plane and completely removing any fixed and predetermined anchoring nodes in the network, while aiming at latency minimization.

The main contribution of this work is a proposal for a mobility-management mechanism, named MAP-Me, with the following characteristics:

- MAP-Me addresses micro (e.g. intra Autonomous Systems) producer mobility. Addressing macro-mobility is a non-goal of this work, left for future work. We are focusing here on complementary mechanisms able to provide a fast and lightweight handover, preserving the performance of flows in progress.
- MAP-Me does not rely on local or global routing updates, which would be too slow and too costly, but rather works at a faster timescale propagating

forwarding updates and leveraging real-time notifications left as breadcrumbs by the producer to enable live tracking of its position[1]. The objective being the support of high-speed mobility and real-time group applications like Periscope [122]. MAP-Me leverages core NDN features like stateful forwarding, dynamic and distributed Interest load balancing to update the forwarding state at routers, and relaying former and current producer locations.

- MAP-Me is designed to be access-agnostic, to cope with highly heterogeneous wireless access and multi-homed/mobile users.
- Low overhead in terms of signaling, additional state at routers, and computational complexity are also targeted in the design to provide a solution able to scale to large and dynamic mobile networks.

To evaluate this proposal, we first contribute an analysis of protocol correctness and guarantees; then, we provide a realistic simulation environment in NDNSim 2.1 [106], where we compare it against an ideal Global Routing, which can instantly and optimally update all FIBs, anchor-based and tracing-based solutions over a set of random waypoint and trace-driven mobility patterns representing V2I scenarios based on 802.11 radio access. Results show that MAP-Me satisfies its objectives while equalling or outperforming the existing alternatives both in terms of user performance (e.g. loss, delays) and network cost (e.g. signaling overhead, link utilization) metrics. We also show wide applicability of results across different topologies radio and mobility patterns.

The rest of the chapter is organized as follows: In Sec. 5.2, we introduce the design principles and details of MAP-Me operations, before analyzing its correctness and path-stretch guarantees in Sec. 5.4. A comprehensive evaluation of the benefits of our proposal is then performed in Sec. 5.5. Finally, Sec. 5.6 investigates the interaction and possible cooperation between MAP-Me and an existing routing protocol, before concluding with the final remarks in Sec. 5.7.

5.2 Design

In this section, we introduce MAP-Me, a micro-mobility management architecture for NDN networks. Based on the classification and discussion made in the previous section, we detail here the design principles inspiring MAP-Me.

[1]For simplicity, we use the word *producer* in place of the more correct expression *producer name prefixes*

- *Transparent:* MAP-Me does not involve any name nor modifications to basic request/reply operations to be compatible with standard NDN design and to avoid issues caused by name modifications like triangular routing, caching degradation, or security vulnerabilities.
- *Distributed:* MAP-Me is designed to be fully distributed, to enhance robustness w.r.t. centralized mobility management proposals subject to single point-of-passage problem.
- *Localised:* MAP-Me updates affect the minimum number of routers at the edge of the network to restore connectivity. The goal is to realize effective traffic off-load close to the end-users.
- *Lightweight:* MAP-Me mobility updates are issued at prefix granularity, rather than content or chunk/packet granularity, to minimize signaling overhead and temporary state kept by in-network nodes.
- *Reactive:* MAP-Me works at forwarding layer to enable updates in FIBs at network latency, i.e. round-trip time scale. Specific mechanisms are defined, referred to as network notifications and discovery, to maximise reactivity in mobility management in case of real-time producer tracking and of latency-sensitive communications.
- *Robust* to network conditions (e.g. routing failure, wireless or congestion losses, and delays), by leveraging hop-by-hop retransmissions of mobility updates.

5.2.1 MAP-Me description

As a data plane protocol, MAP-Me handles producer mobility events by means of dynamic FIB updates with the objective of minimizing unreachability of the producer. It relies on the existence of a routing protocol responsible for creating/updating the FIB of all routers, possibly with multipath routes, and for managing network failures (eg. [164, 65]). MAP-Me is composed of:
- an Update protocol (MAP-Me-IU) (Sec.5.2.2), which is the central component of our proposal;
- a Notification/Discovery protocol (Sec.5.2.3), to be coupled with the Update protocol (the full approach is referred to as MAP-Me) to enhance reactivity in mobility management for realtime/latency-sensitive application.

In this section, we describe the protocols independently of the routing protocol, and dedicate Sec. 5.6 at their integration. Routing will be in charge of handing the long-term impact of mobility (node relocation), and reoptimizing paths. We

assume a single producer of content, and no prefix reaggregation at intermediary ISPs.

5.2.2 MAP-Me Update protocol

5.2.2.1 Rationale

The rationale behind MAP-Me-IU is that the producer announces its movements to the network by sending a special Interest Packet, named *Interest Update* (IU) to "itself" after it reattaches to the network. Such a message looks like a regular Interest packet named with the prefix advertised by the producer. As such, it is forwarded according to the information stored in the FIBs of traversed routers towards previous locations of the producer known by router FIBs. A special flag carried in the header of the IU enables all routers on the path to identify the Interest as a mobility update and to process it accordingly to update their FIBs (a detailed description of the IU processing is provided in Sec.5.3.2).

The key aspect of the proposal is that it removes the need for a stable home address (present in Tracing-Based approaches for instance) by directly leveraging name-based forwarding state created by NDN routing protocols or left by previous mobility updates. FIB updates are triggered by the reception of mobility updates in a fully distributed way and allow a modification on-the-fly to point to the latest known location of the producer.

5.2.2.2 Updates propagation

MAP-Me-IU aims at quickly restoring global reachability of mobile prefixes with low signaling overhead, while introducing a bounded maximum path stretch (i.e. ratio between the selected and the shortest path in terms of hops). Let us illustrate its behavior through the example in Fig. 5.1, where a single producer serving prefix p moves from position P_0 to P_1 and so on. **Fig. 5.1(a)** shows the tree formed by the forwarding paths to the name prefix p where IU initiated by the producer propagates.

Network FIBs are assumed to be populated with routes toward P_0 by a name-based routing protocol (yellow cloud) After the relocation of the producer from P_0 to P_1, once the layer-2 attachment is completed, the producer issues an IU carrying the prefix p and this is forwarded by the network toward P_0 (in general, toward one of its previous locations according to the FIB state of the traversed routers).

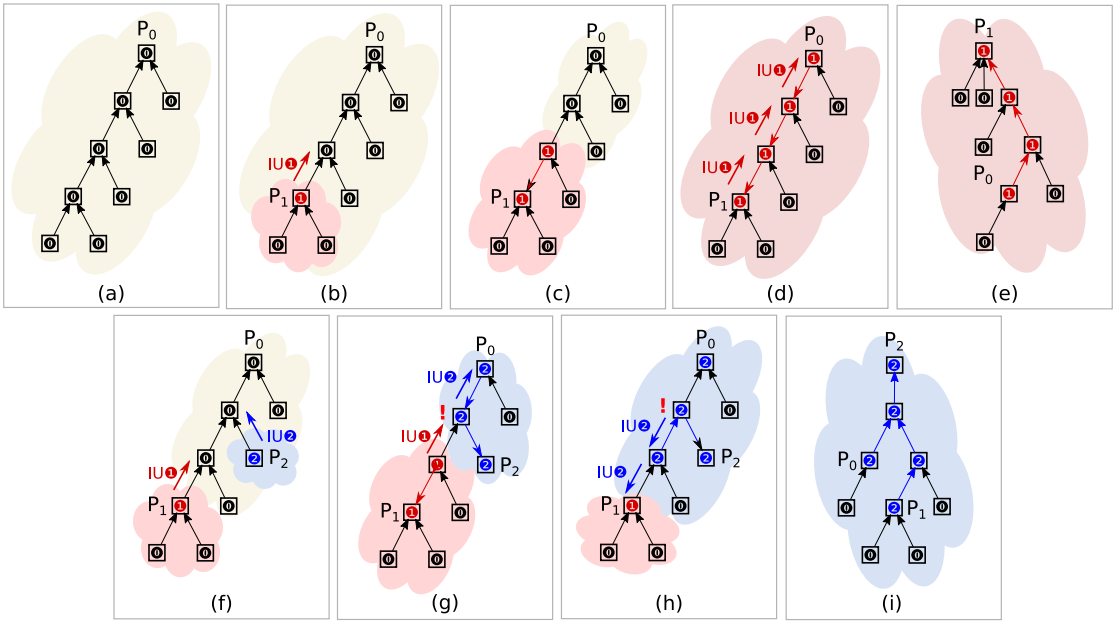


FIGURE 5.1: MAP-Me-IU illustration.

Fig. 5.1(b) shows the propagation of the IU. As the IU progresses, FIBs at intermediate hops are updated with the ingress face of the IU (Fig 5.1(c, d)). IU propagation stops when the IU reaches P_0 and there is no next hop to forward it. The result is that the original tree rooted in P_0 becomes re-rooted in P_1 (Fig. 5.1(e)). Looking at the different connected regions, we see that IU propagation and consequent FIB updates have the effect of extending the newly connected subtree (represented as a red cloud) : at every step, an additional router and its predecessors are included in the connected subtree. We will analyze the properties of the update propagation process in terms of bounded lenght and stretch in Sec.5.4.

5.2.2.3 Concurrent updates

Frequent mobility of the producer may lead to the propagation of concurrent updates. To prevent inconsistencies in FIB updates, MAP-Me-IU maintains a sequence number at the producer end that increases at each handover and identifies every IU packet. Network routers also keep track of such sequence number in FIB to verify IU freshness. Without detailing the specific operations in MAP-Me to guarantee update consistency (whose description is provided in Sec.5.3.2), we can say that modification of FIB entries is only triggered when the received IU carries a higher sequence number than the one locally stored, while the reception of a less recent update determines a propagation of a more recent update through the not-yet-updated path. An example of reconciliation of

concurrent updates is illustrated in Figure 5.1(f), when the producer has moved successively to P_1 and then to P_2 before the first update is completed.

Both updates propagate concurrently until the update with sequence number 1 (IU_1) crosses a router that has been updated with fresher information – that has received IU with higher sequence number (IU_2) as in Fig. 5.1(g). In this case, the router stops the propagation of IU_1 and sends back along its path a new IU with an updated sequence number (Fig. 5.1(h)). The update proceeds until ultimately the whole network has converged towards P_2 (Fig. 5.1(i)).

MAP-Me-IU protocol reacts at a faster timescale than routing – allowing more frequent and numerous mobility events – and over a localized portion of the network edge between current and previous producer locations. We thus expect MAP-Me-IU respectively to minimize disconnectivity time and to reduce the link load, which are the main factors affecting user flow performance, as later shown by our evaluation in Section 5.5.

5.2.3 Map-Me Notification/Discovery protocol

IU propagation in the data plane accelerates forwarding state re-convergence w.r.t. routing-based (RT) or resolution-based (RB) approaches operating at control plane, and w.r.t. anchor-based (AB) approaches requiring traffic tunneling through the anchor. Still, network latency makes IU completion not instantaneous and before an update completes, it may happen that a portion of the traffic is forwarded to the previous PoA and dropped because of the absence of a valid output face leading to the producer.

Previous work in the Local Routing category has suggested the buffering of Interests at previous producer location [77] to prevent such losses by increasing network reactivity. However, such a solution is not suitable for applications with stringent latency requirements (e.g. real-time) and may be incompatible with IU completion times. Moreover, the negative effects on latency performance might be further exacerbated by IU losses and consequent retransmissions in case of wireless medium. To alleviate such issues, we introduce two separate enhancements to MAP-Me-IU protocol, namely *(i)* an **Interest Notification** mechanism for frequent, yet lightweight, signaling of producer movements to the network and *(ii)* a scoped **Producer Discovery** mechanism for consumer requests to proactively search for the producer's recently visited locations.

5.2.3.1 Interest notification

An Interest Notification (IN) is a breadcrumb left by producers at every encountered PoA. It looks like a normal Interest packet carrying a special identification flag and a sequence number, like IUs. Both IU and IN share the same sequence number (producers indistinctly increase it for every sent message) and follow the same FIB lookup and update processes. However, unlike IU packets, the trace left by INs at the first hop router does not propagate further. It is rather used by the discovery process to route consumer requests to the producer even before an update process is completed.

It is worth observing that updates and notifications serve the same purpose of informing the network of a producer movement. The IU process restores connectivity and as such has higher latency/signaling cost than the IN process, due to message propagation. The IN process provides information to track producer movements before update completion when coupled with a scoped discovery. The combination of both IU and IN allows to control the trade-off between protocol reactivity and stability of forwarding re-convergence.

5.2.3.2 Discovery

The extension of MAP-Me with notifications relies on a local discovery phase: when a consumer Interest reaches a PoA with no valid output face in the corresponding entry, the Interest is tagged with a "discovery" flag and labeled with the latest sequence number stored in FIB (to avoid loops). From that point on, it is broadcasted with hop limit equal to one to all neighbors and discarded unless it finds the breadcrumbs left by the producer to track him (notifications). The notifications can either allow to forward consumer Interests directly to the producer or give rise to a repeated broadcast in case of no valid output face. The latter is the case of a breadcrumb left by the producer without any associated forwarding information because the producer has already left that PoA as well. A detailed description of the process is reported in Sec.5.3.2.

As further shown in Sec. 5.5, the notification/discovery mechanism is important to preserve the performance of flows in progress, especially when latency-sensitive.

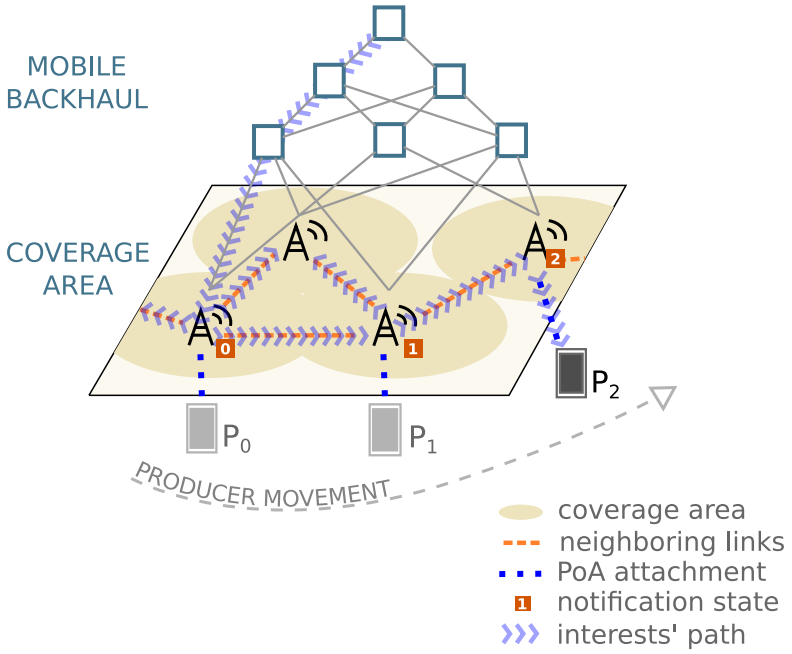


FIGURE 5.2: Notifications/Discovery process example.

5.2.4 Full MAP-Me approach

In the rest of the chapter, we evaluate a combined update and notification/discovery approach consisting of sending a IN immediately after an attachment and a IU at most every T_U seconds, referred to as MAP-Me, to reduce signaling overhead especially in case of high mobility. The update-only proposal, denoted as MAP-Me-IU, is also evaluated separately.

Figure 5.2 illustrates the combined use of notifications and discovery in a mobile access network where the different PoAs form the leaves of a fat tree.

The producer, initially in position P_0 moves to P_1 and later to P_2, sending each time an Interest Notification (respectively IN 1 and IN 2. Consumer interests are forwarded using FIB information synchronized with the initial state of the producer and thus reach the initial PoA, P_0. Once the producer moves and the face is destroyed, no valid next-hop face information can be found into the FIB and consumer Interests reaching P_0 enter in discovery mode: they are tagged with the sequence number 0 found in the FIB, and broadcasted to one-hop neighbors, which may either forward them directly to the producer (this is the case for the current PoA of the producer) or broadcast them one hop further if they have been notified of producer attachment by means of INs, but there is no valid forwarding information. Other network nodes reached by a Discovery Interest just discard the packet since they have no fresher information about the position of the producer. The discovery process is iterated until the producer is reached.

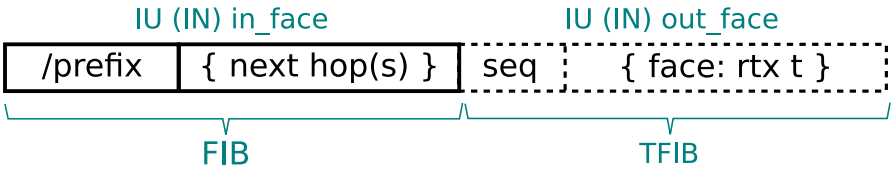


FIGURE 5.3: MAP-Me FIB/TFIB description.

5.3 Implementation

5.3.1 MAP-Me introduction in a NDN network

In this section we describe the changes to a regular NDN architecture required to implement MAP-Me and detail the above-described algorithms. This requires to specify a special Interest message, additional temporary information associated to the FIB entry and additional operations to update such entry.

5.3.1.1 MAP-Me Messages

Two new optional fields are introduced in a NDN Interest header:

- a special *Interest Type* (T) to specify four types of messages: Interest Updates (IU), Interest Notifications (IN), as well as their associated acknowledgment (Ack) messages (IU_{ack} and IN_{ack}). Those flags are recognized by the forwarding pipeline to trigger special treatment.
- a *sequence number*: used to handle concurrent updates and prevent forwarding loops during signaling, and to control discovery interests during consumer interest propagation;

5.3.1.2 MAP-Me additional Network Information

FIB entries are enriched with a sequence number, set to 0 say, by routing protocol updates and updated by MAP-Me upon reception of IU/IN messages. The Data about not-yet-acknowledged messages are temporarily stored in what we denote as **Temporary FIB buffer, TFIB,** to ensure the reliability and robustness of the process, and removed upon reception of the corresponding acknowledgement. As sketched in Fig.5.3, each TFIB entry is composed of an associative array ($F \rightarrow T$) mapping a face F on which IU has been sent with the associated retransmission timer T (possibly null, denoted $\perp$).

Observe that the update mechanism is a constant delay operation at each router/hop and is performed at line rate.

5.3.2 Algorithm description

5.3.2.1 IU/IN transmission at producer

MAP-Me operations are triggered by producer mobility, i.e. after detaching from a network node and reattaching to a new one. At the producer end, a mobility event is followed by a layer-2 attachment and, at network layer, a change in the FIB. More precisely, a new face is created and activated upon attachment to a new base station. This signal triggers the increase of MAP-Me sequence number and the transmission of an IU (or IN message depending on the selected MAP-Me mechanism) for every served prefix carrying the computed sequence number.

To ensure reliable delivery of IUs, a timer is setup in the temporary section of the FIB entry (TFIB). If an acknowledgement of the IU/IN reception is not received within τ seconds since the packet transmission, a retransmission of IU is rescheduled.

We define the `SendReliably(`F`,` *type*`,` ϵ`)` function for sending Special Interests of type *type* on faces F based on name and sequence number found in local FIB entry ϵ. It schedules their retransmission through a timer T stored in TFIB: $\epsilon.\text{TFIB} = \epsilon.\text{TFIB} \cup (F \rightarrow T)$, and removed on Ack.

5.3.2.2 IU/IN processing at network routers

At the reception of IU/IN packets, each router performs a name-based Longest Prefix Match lookup in FIB to compare IU/IN carried and FIB stored sequence number. According to the result of the comparison:
- if the IU/IN packet carries a higher sequence number, the existing next hops associated to the lower sequence number in FIB are used to forward further the IU (INs are not propagated) and temporarily copied into TFIB to avoid loss of such information before completion of the IU/IN acknowledgement process (in case of IN, such entries in TFIB are set with a $\perp$ timer to maintain a trace of the producer recent attachment). Also, the originating face of the IU/IN is added to FIB to route consumer requests to the latest known location of the producer.
- If the IU/IN packet carries the same sequence number as in the FIB, the originating face of the IU/IN is added to the existing ones in FIB without additional packet processing or propagation. This may occur in presence of multiple forwarding paths.
- If the IU/IN packet carries a lower sequence number than the one in the FIB, FIB entry is not updated as it already stores "fresher information". To advertise the latest update through the path followed by the IU/IN packet, this one is re-sent through the originating face after having updated its sequence number with the value stored in FIB.

5.3.2.3 Hop-by-hop IU/IN acknowledgement

The operations in the forwarding pipeline for IU/IN processing are reported in Algorithm 2.

Algorithm 2: ForwardSpecialInterest(SpecialInterest SI, Ingress face F)

```
CheckValidity()
▷Retrieve the FIB entry associated to the prefix
ϵ, T ← FIB.LongestPrefixMatch(SI.name)
if SI.seq ≥ ϵ.seq then
    ▷Acknowledge reception
    s ← ϵ.seq
    ϵ.seq ← SI.seq
    SendReliably(F, SI.type + Ack, ϵ)
    ▷Process special interest
    if F ∈ ϵ.TFIB then
        ▷Remove outdated TFIB entry (eventually cancelling timer)
        ϵ.TFIB = ϵ.TFIB \ F
    if SI.seq > s then
        if SI.type = IU then
            ▷Forward the IU following FIB entry
            SendReliably(ϵ.NextHops, SI.type, ϵ)
        else
            ▷Create breadcrumb and preserve forwarding structure
            ϵ.TFIB = ϵ.TFIB ∪ {(f → ⊥) : ∀f ∈ ϵ.NextHops}
        ϵ.NextHops = ∅
    ϵ.NextHops = ϵ.NextHops ∪ F
else
    ▷Send updated IU backwards
    SI.seq = ϵ.seq
    SendReliably(F, SI.type, ϵ)
```

5.3.2.4 Face removal at producer/network nodes

Upon producer departure from a base station, the corresponding face is destroyed. If this leads to the removal of the last next hop, then faces in TFIB with $\perp$ timer (entries generated by notifications) are restored in FIB to preserve the original forwarding tree and thus global connectivity.

5.3.2.5 Consumer request forwarding in case of producer discovery

The forwarding of regular Interests is mostly unaffected in MAP-Me, except in the case of discovery Interests that we detail in Algorithm 3). The function `SendToNeighbors(`I`)` is responsible for broadcasting the Interest I to all neighboring PoAs.

When an Interest arrives to a PoA which has no valid next hop for it (because the producer left and the face got destroyed), it enters a discovery phase where the Interest is flagged as a Discovery Interest and with the local sequence number, then broadcasted to neighboring PoAs to track the producer.

Upon reception of a Discovery Interest, the PoA forwards it direcly to the producer if still attached, otherwise it repeats the one-hop brodcast discovery to neighboring PoAs if it stores a recent notification of the producer presence, i.e. an entry in TFIB with $\perp$ timer and higher sequence number than the one in the Discovery Interest. Otherwise, the Discovery Interest is simply discarded.

It is worth observing that the discovery process is initiated only in the case of no valid forwarding next hop and not every time a notification is found in a traversed router. This is important to guarantee that the notification/discovery process does not affect IU propagation and IU process completion.

5.3.3 Security considerations

As all mobility solutions, MAP-Me affects the forwarding of user traffic and its updates have to be secured. An in-depth study will be the subject of future work, and we restrict our ambition here to demonstrate how the existing solutions from the literature can provide an equivalent security as the current approaches used today in operational networks. It is indeed worth noticing that although our approach seems disruptive, it shares many similarities with the existing macro or micro-mobility solutions. In Anchor-Based approaches for instance (say M-IPv6), it is also up to the producer to issue registration requests to an agent to update the binding between the foreign and the care-of-address (tunnels), which also affects end-to-end forwarding.

The general approach is to distinguish a boostrap procedure when a node enters the domain, in which we typically use PKI-based schemes to do comprehensive vetting of

Algorithm 3: InterestForward(Interest I, Origin face F)

```
▷Regular CS and PIT lookup
ϵ ← FIB.LongestPrefixMatch(I.name)
if ϵ = ø then
  | return
if I.seq = ø then
  | ▷Regular interest
  | if hasValidFace(ϵ.NextHops) or DiscoveryDisabled then
  |   | ForwardingStrategy.process(I, ϵ)
  | else
  |   | ▷Enter discovery mode
  |   | I.seq ← ϵ.seq
  |   | SendToNeighbors(I)
else
  | ▷Discovery interest: forward if producer is connected...
  | if hasProducerFace(ϵ.NextHops) then
  |   | ForwardingStrategy.process(I, ϵ)
  | ▷...otherwise iterate iif higher seq and breadcrumb
  | else if ϵ.seq ≥ I.seq ∧ ∃f|(f → ⊥) ∈ ϵ.TFIB then
  |   | I.seq ← ϵ.seq
  |   | SendToNeighbors(I)
```

the users' authorizations. It is then possible to use a secure token to quickly authorize subsequent calls from a user, e.g. by using an HMAC function.

Among the many variations that can be found in the literature, one such example is given by the authors of Cellular IP [36] in a distributed context that is close enough to MAP-Me to be readily applicable. We limit our description here to the bare mechanism, which can be further secured through the use of random identifiers in HMAC computation.

Assume that all infrastructure nodes share a symmetric key κ. Upon bootstrap, a producer willing to announce prefix p will receive a derived key $\kappa_U = \text{HMAC}_\kappa(p)$ along with random number r.

While all network nodes can compute such a key from the information given by the producer, it won't allow the user to retrieve the network key. Signaling message can then be augmented with a registration token $R = \text{HMAC}_{\kappa_U}(p \| seq)$, where $\|$ means concatenation. R can be verified by all nodes to prove that the user had indeed been authorized during a previous bootstrap. Indeed, all nodes can compute K_U then R based on the information provided by the producer in the special interest (while the producer cannot of course get back the network key).

5.4 Analysis

In this section, we investigate MAP-Me guarantees of forwarding update correctness and path stretch stability and we support them by numerical evaluation over known ISP network topologies. For the sake of clarity, the analysis reports the formal proofs only in case of single-path routing.
The extension to multipath routing is straightforward by using directed acyclic graphs instead of trees.

5.4.1 Correctness and stability of IU mechanism

We consider m consecutive movements of the producer in network positions $\{P_0, P_1, ..., P_m\}$ and focus on forwarding state variations determined by MAP-Me at the time instants corresponding to either producer movements or Interest Update processing. We observe that at any such instant, as illustrated in Fig.5.1, the network is partitioned into a set of islands, whose number varies in $[1, m+1]$ as a function of producer movements and, hence, of the number of ongoing update processes. We assume that global routing guarantees the existence of a spanning tree (SP) rooted in the original location P_0 at the beginning of the mobility process. The tree is not required to be a minimum SP or a shortest-path tree. About the completion of the update process after a given movement k, we can state that:

Proposition 5.1. *MAP-Me update mechanism guarantees finite completion time of update k, $\forall k \in [1, m]$ in a bounded number of hops equal to* $2\left(\max_{0 \leq j < k}(|P_k - P_j| - 1)\right)$*;*

Proof. Assuming that IU losses are handled by the retransmission mechanism described in Sec.5.2, the hop-by-hop propagation of an IU has two possible outcomes: either (i) the next router has a sequence number, which is inferior to the IU carried sequence number; in this case, IU continues its propagation towards the root of the latest routed tree, decreased by 1 hop; or (ii) the router has a more recent sequence number, hence the IU is sent back with the encountered higher sequence number towards the originating routed position of the producer. Since the maximum sequence number is bounded by m, the maximum number of hops traversed by IU with sequence number k is finite.

More precisely, the maximum number of hops traversed by IU with sequence number k, IU_k is bounded by twice the maximum distance between the originating router P_k and the farthest previous location P_j, $j < k$ minus one, i.e. $2\left(\max_{0 \leq j < k}(|P_k - P_j| - 1)\right)$. Indeed, the worst case occurs when IU_k encounters a more recent update $k' > k$ at the hop before reaching the latest routed previous location, which can also coincide with the farthest one in terms of distance. In such a case, IU_k propagates back to P_k carrying k' sequence number before stopping. □

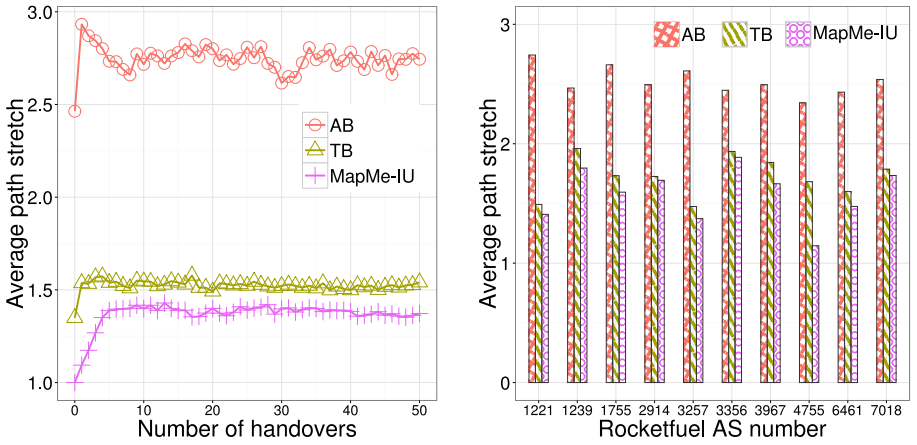
After IU_k propagation, the router P_k and all its predecessors traversed by IU_k to reach the last routed location are connected to the island of highest encountered sequence number, and thus the number of distinct islands is reduced by one unit. By iterating the same process on all IUs, it is straightforward to see that at IU_m completion $m+1$ islands associated to sequence number $0, 1, ..., m-1$ will have merged into the island created by IU_m. With regard to the properties of an island, we can state the following.

Proposition 5.2. *Given a sequence of m consecutive movements of producer position on the routing tree rooted in P_0, producer movement m induces a new tree rooted in P_m.*

Proof. The initial spanning tree is a rooted directed tree in P_0 giving the routes to reach all nodes in the network. MAP-Me update mechanism after movement m flips all directed links from P_m to the latest routed position P_j, $j < m$, so that they point to P_m. In the presence of multiple concurrent updates, the most recent one, i.e. the one with the highest sequence number, also propagates back along the routes of the encountered previous updates. Thus, update completion results in fully merging different rooted trees into the one of highest sequence number, m, rooted in P_m. □

Corollary 5.3. *MAP-Me is loop-free under loop-free global routing.*

Proof. Starting from the spanning tree given by global routing, Prop.5.2 states that MAP-Me induces a new tree, as it only flips all edges over the unique path from the



(a) AS 1221 stretch evolution

(b) Stretch comparison

FIGURE 5.4: Path stretch evolution (a) and comparison (b) over Rocketfuel topologies.

original position to the new one. Indeed, given the unchanged number of links/nodes, the result is still a directed tree rooted in the new position. Hence, it is loop-free. □

Proposition 5.4. *MAP-Me path stretch for node i over the tree rooted in P_m, created after producer's m-th movement, is upper bounded by the ratio $(|i-P_0|_{P_0}+|P_0-P_m|_{P_0})/|i-P_m|_{P_m}$ as $m \to \infty$, which corresponds to the path stretch of the anchor-based approach with anchor in P_0.*

Proof. We can distinguish two cases according to whether P_0 is on the path between i and P_m on the P_m-rooted tree or not. If it is, then the path between i and P_m may be split into the paths i to P_0 and P_0 to P_m. The second component is equal to the path length between P_m and P_0 on the initial tree (only directions have been flipped).

The first one corresponds to the same path on the initial tree even in terms of directions. Therefore, the path stretch in this case is exactly equal to $(|i-P_0|_{P_0}+|P_0-P_m|_{P_0})/|i-P_m|_{P_m}$. Otherwise, if P_0 is not on the path between i and P_m, the path between i and P_m is, by definition of MAP-Me update process (that utilizes the shortest path routing for IUs), shorter than the one including the detour via P_0 on the initial P_0-rooted tree. The bound remains true as $m \to \infty$, because it is intrinsically related to the properties of the initial tree. □

5.4.2 Numerical Evaluation of path stretch

We compute now the average path stretch obtained by AB, TB, and MAP-Me-IU (MAP-Me requires geographical mobility and will be later considered in Section 5.5), on the

topologies obtained from the Rocketfuel project[2]. The initial position on the consumer, producer, and eventual anchor are chosen randomly, and two mobility models are implemented: (i) uniform – like in the related work – where the producer can jump towards any other node from the graph, and (ii) random waypoint (RWP), where the producer chooses a waypoint like in the previous approach, but advances hop-by-hop on the shortest path towards that waypoint, and then starts again. We average over 1000 runs to compute ensemble average for the path stretch after k movements of the producer with small confidence intervals. Figure 5.4(a) represents the evolution of average MAP-Me path stretch over AS 1221 topology under RWP (other patterns show similar trends). We observe that path stretch stabilizes beyond 10 movements, because MAP-Me preserves the initial structure of the forwarding tree (it only modifies links direction). Other Rocketfuel topologies show quantitatively the same results. A comparison of the different approaches over the all 10 Rocketfuel topologies is in Fig.5.4(b) and 5.4(b). Under both uniform and RWP mobility, MAP-Me outperforms AB, achieving up to 55% stretch reduction, as well as TB.

5.5 Evaluation

5.5.1 Simulation setup

The section gathers simulation assessment of MAP-Me over different mobility pattern, radio conditions and network topologies. We implemented both MAP-Me and MAP-Me-IU, anchor-based (AB), tracing-based (TB) – based on Kite ([181]) – and routing-based GlobalRouting (GR) – an idealized routing plane instantly updating all routers – approaches in NFD within the NDNSim 2.1 framework. We don't consider here resolution-based (RB) or other LR solutions as they are not appropriate for latency-sensitive applications (see discussion in Sec. 2.3)

We first evaluate all mobility protocols in a baseline scenario, before varying parameters such as radio conditions, mobility model and network topology in order to gain insight into their sensitivity. All plot data is averaged over many runs, or a large number of handover(at least 250 per mobile node per run) depending on the context; although, for clarity, we chose not to display confidence intervals. The full set of results is available in [21].

[2] `http://research.cs.washington.edu/networking/rocketfuel/`. We extract the undirected graph corresponding to the largest connected component as in [166, 181]. We remark that the authors either did not use the same data, or processed the graph in an undocumented way, which prevents us to reproduce their results. We still obtain similar qualitative conclusions.

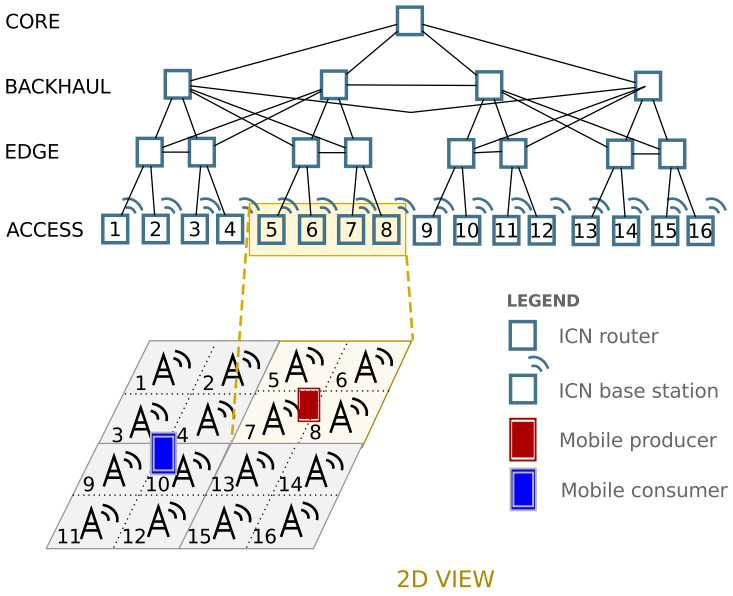


FIGURE 5.5: Network with link capacity C=10Mb/s.

5.5.2 Baseline scenario description

Topology: In the baseline scenario, we use 802.11n access network composed of a 4-by-4 grid of base stations (BS) with square-shaped cell of side $s = 80m$. They are connected to a fat-tree backhaul network represented in Fig. 5.5. This choice is motivated by the similarity in terms of redundancy and meshing found in real ISP access network. Wired links have a capacity of C=10Mb/s and 5ms delay. We complement it by a wide range of well-known topology models or Rocketfuel topologies to cover all types of graph metrics in the variant of baseline of scenario in section 5.5.4.

Radio and Mobility: We use IEEE 802.11n WiFi on 5GHz frequencies, with Minstrel rate adaptation [108] and log distance propagation model plus Rayleigh-fading model for wireless channel. Mobile nodes move in the 4x4 cells under full radio coverage. We choose random waypoint mobility model for user mobility. We also vary mobile's moving speed from 1m/s to 15m/s (i.e, pedestrian to vehicular speed). A range of other radio propagation models and mobility models are also used in the variant scenario in section 5.5.4.

Application: We assume N disjoint pairs of mobile consumers and producers. In particular, we choose N=5 for baseline scenario and also its variants. To highlight MAP-Me benefits in the support of latency-sensitive traffic, we consider a streaming audio/video application, characterized by a CBR rate of 1Mbps without retransmission in the baseline scenario, and further extend it with an adaptive protocol inspired by the Periscope streaming application in Sec. 5.5.6. While, the CBR application has the nice property of reflecting network performance, the adaptive one has a closed-loop behavior that is more realistic but might be affected by wireless and mobility losses. More in-depth study of these interactions is out of scope for this work.

5.5.3 Results for baseline scenario: Fat-Tree + RWP + CBR

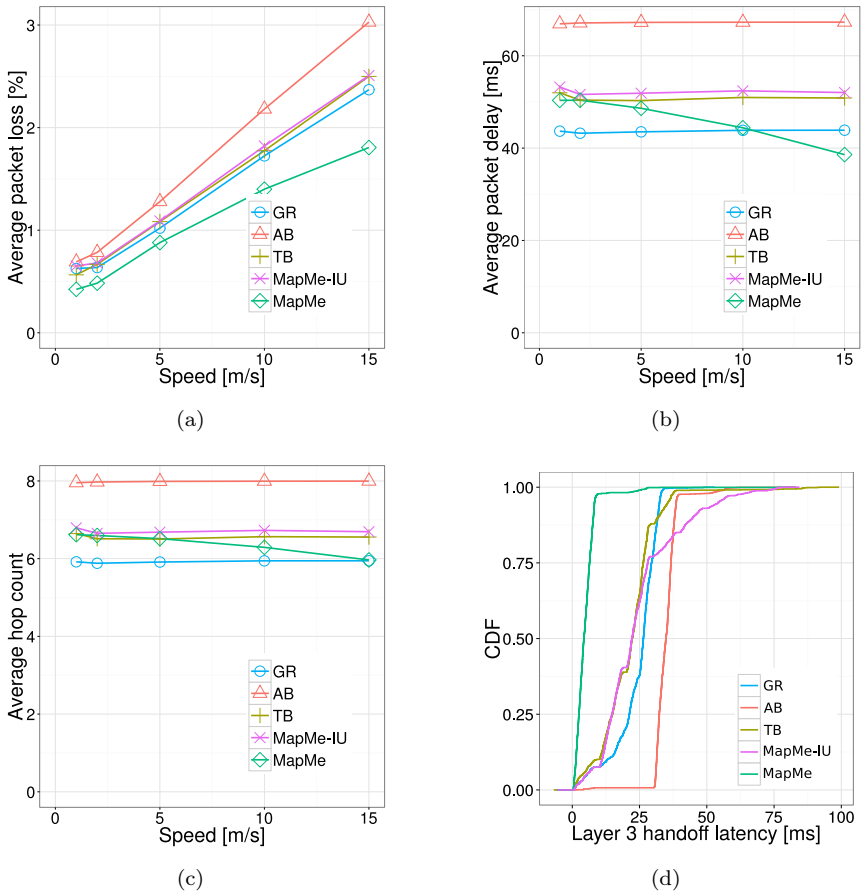


FIGURE 5.6: User performance: packet loss (a), delay (b), and hop count (c); CDF L3 hand-off latency (d).

5.5.3.1 User performance

In Fig.5.6(a)-5.6(b), we show two performance indicators for latency-sensitive traffic, average packet loss and delay, in case of $N = 5$ consumer/producer pairs as a function of mobile speed (from pedestrian, i.e. 1m/s or 3km/h to vehicular, i.e. 15m/s or 54km/h). We can distinguish two kinds of losses: due to the wireless medium, occurring irrespective of the mobility management approach, and those due to mobility. The fraction of mobility losses is consistently reduced by MAP-Me, especially in the presence of the notification/discovery mechanism, as a result of in-fly re-routing of Interests towards the new location of the producer, which prevents Interest timeouts. MAP-Me-IU like TB (or alternative LR solutions) enables re-routing of Interests only after the interval of time required for an update to complete. A longer time is required for a global routing update, but the resulting path is the shortest possible, which explains the equivalent performance w.r.t. MAP-Me-IU/TB. AB under performs because of worse update completion time and path stretch. The experienced average packet delay in Fig.5.6(b) is a consequence of

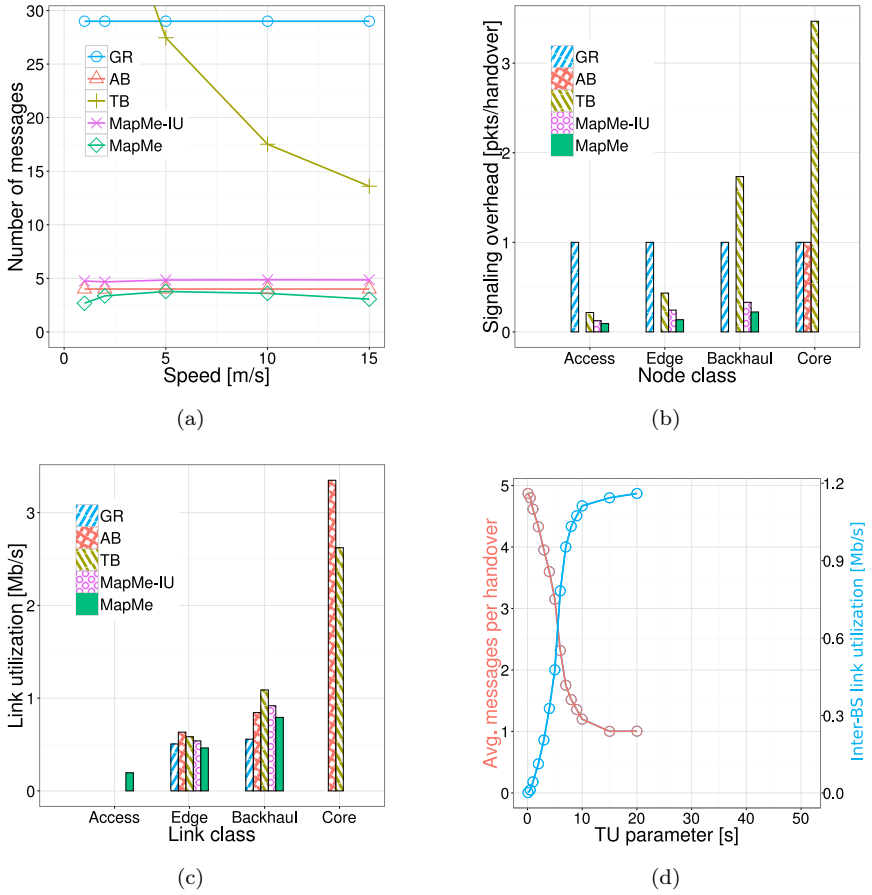


FIGURE 5.7: Network cost: Signaling overhead vs mobile speed (a), overhead (b), and link utilization (c) per router class. Map-Me sensitivity analysis (d).

the path stretch of different approaches: high for AB, medium for TB or MAP-Me-IU, low for GR. MAP-Me achieves better performance especially at high speed when the discovery/notification mechanism is mostly used in virtue of the shorter 1-hop forwarding between APs at the access that does not involve upper links in the topology (at the edge level). As explained, packet losses and delay result from the different average path lengths associated to each mobility update process, see Fig.5.6(c), and from the L3 hand-off latency, i.e. the time required for L3 reconnection after a handover, see Fig.5.6(d). The L3 hand-off latency illustrates the reactivity of the mobility-management protocol and highlights the significant improvement brought by MAP-Me, which reduces latency to zero. It is interesting to observe that AB shows a constant latency value of around 30ms due to update propagation up to the anchor, while for GR, TB, and MAP-Me-IU, such latency varies according to the number of routers to be updated, as a function of producer movement in the considered topology. Latency variations can be visualized at the inflection points in the corresponding CDFs in Fig.5.6(d).

5.5.3.2 Network cost

If user performance is critical to drive mobility-management choice, network cost analysis is equally important for the selection of a cost-effective solution. To this aim, we compare signaling overhead, meaning the total number of control messages triggered by a handover, in Fig.5.7(a), and the volume of signaling messages per handover to be processed by routers at different positions in the network, in Fig.5.7(b). More precisely, in the latter case, we visualize the distribution over the network of signaling load by distinguishing the average number of messages per handover received by different classes of routers, based on their position in the network: access, edge, backhaul, core as indicated in Fig.5.5. As expected, the overall number of signaling messages as a function of mobile speed is constant for AB, equal to the number of hops from mobile nodes to the anchor (4). Instead, it varies for MAP-Me and MAP-Me-IU according to the also varying average hop count (i.e. path stretch), as already observed in Fig.5.6(c). TB approaches involve a much higher signaling overhead due to "keep-alive" messages periodically sent to refresh update information. By reporting the way traffic is spread across the network and where signaling traffic goes, we can draw some key observations. Every mobility protocol relies on the control plane that enforces a routing state across the network (shortest-path routing in this chapter), which corresponds to the initialization state for mobility. All protocols relying on a fixed anchor have routing pointing to the anchor's location, whereas for LR mechanisms, it points to the producer's position at the routing update time. Thus, LR approaches are able to offload mobile backhaul and core networks from all local traffic, seamlessly (Fig.5.7(c)[3]).

Finally, we report about MAP-Me sensitivity to parametrization, i.e. the impact of TU settings. In Fig.5.7(d), we observe that MAP-Me has robust parametrization as long as TU is not too small (signaling overhead and path stretch quickly converges to the best settings) or too high (load on access).

5.5.4 Impact of mobility pattern, radio conditions and topology

We have performed extensive simulations to evaluate the sensitivity of MAP-Me and other solutions, by varying several parameters in our baseline scenario [21]. We report here our most significant results and confirm the wide applicability of conclusions from the previous section.

[3]For clarity, utilization of access link only represents traffic between base stations, excluding upstream from mobiles.

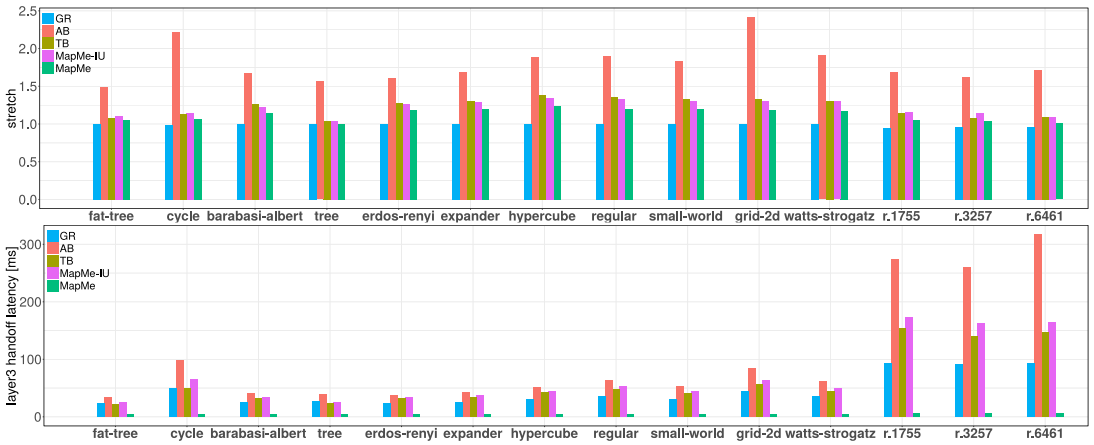


FIGURE 5.8: Path stretch and handoff latency for simulated network topologies (r.1755, r.3257, r.6461 are Rocketfuel topologies).

5.5.4.1 Impact of mobility pattern and radio conditions

For mobility patterns, we have included the previous jump models across base-stations and classical models available in NS-3 (Random Direction 2D, Gauss-Markov and Random-Walk 2D models). For radio conditions, we have considered an ideal wireless channel (no loss nor interference at layer 2) by dynamically switching wired links up and down to emulate mobile handover across base stations, and the two radio models from ITU specifications [70], namely urban environment without line of sight (LoS), and suburban with LoS.

The impact of both radio and mobility patterns are negligible, and the plots show no significant difference between performance metrics. Comparative simulations with the ideal wireless channel (not represented here) show that the loss rate is not only due to the wireless channel, but also impacted by the mobility scheme in place, and more specifically the time to reestablish connectivity at L3 (denoted L3 handover time). Moreover, with ideal wireless channel where we can extract out the impact of only L3, we see the relative order of performance of protocols are the same as those in Fig 5.6(a), confirming MAP-Me's superior performance in reducing mobility losses.

5.5.4.2 Impact of topology

We cover a wide range of network characteristics through the use of deterministic and stochastic graphs drawn from well-known models [21], as well as previously described Rocketfuel topologies. While not being representative of access networks, they provide insights into the performance of MAP-Me-IU in non-local mobility (e.g. jumps from WiFi to LTE networks). Edges nodes are randomly picked from graph nodes (or Rocketfuel leaves) to be connected to the previously described grid, while others form the backhaul.

As expected, topology is the most impacting parameter for absolute performance, being the direct consequence of the forwarding trees built atop. Figure 5.8 shows path stretch and L3 handoff latency. As hinted in Sec. 5.4.2 and shown in previous simulations, we see that MAP-Me-IU and TB both offer lower stretch than AB – sometimes close to optimum (GR) – with a slight advantage to MAP-Me-IU in almost all cases. Those variations can be interpreted as the ability for the spanning tree (shortest path tree rooted in anchor for TB, and first producer location for MAP-Me-IU) to offer short paths between consumer and producer, and thus offload traffic at the edge.

Available path lengths are reflected in the CDF of Layer 3 handover latency, and we see that MAP-Me-IU is able to find shorter paths for close-by nodes (effectively offloading traffic), while those towards remote nodes are less optimal than if we were going through the anchor like in TB (hence the crossing of both curves). The reason is we had to enable all optional extensions in TB as in the default setting some situations could lead to unreachability of the producer. One consists in duplicating interests both on the trace and towards the anchor, which had the side effect on selecting the shortest of the two paths. This simple alternative is the reason of the much lower delays observed in TB.

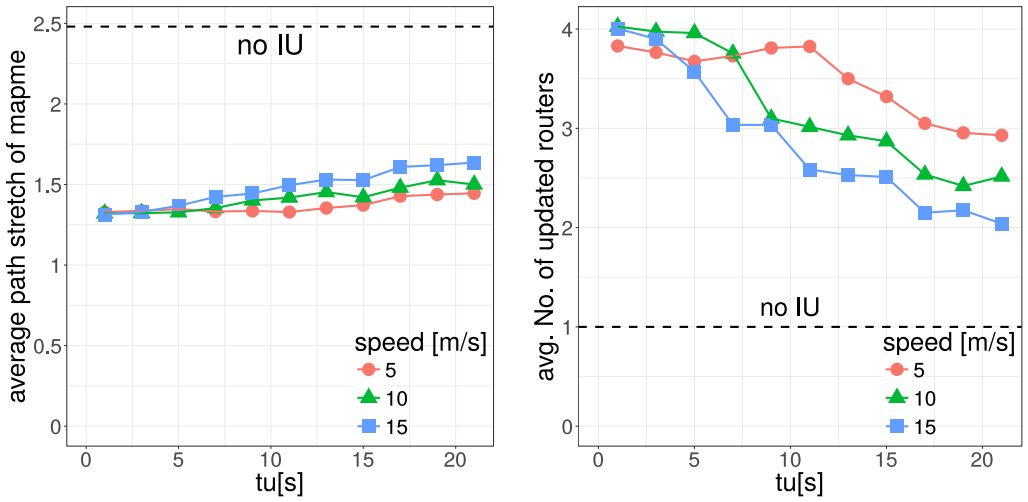
In all cases, we see the extremely low handoff delay ensured by MAP-Me, which confirms the benefits of notification to reduce the time the producer is disconnected, and thus support latency-sensitive applications during mobility.

Beyond confirming our previous observations, these simulations open the way to further extensions of MAP-Me by considering how an alternative routing might lead to better performance – for instance using more efficient spanning trees (ST) such as minimum diameter ST (see Prop.3 in Sec. 5.4.1) – and how more appropriate graph spanners and random strategies could allow the exploration of more than one path.

5.5.5 Impact of notifications on path stretch

As we have seen, the use of notifications improves performance during fast mobility by using inter-PoA links with the risk of increasing path stretch. We show here that the use of T_U as per the selected mechanism (Sec. 5.2.4) changes the root of the IN's breadcrumb chain and thus limits its length. We thus evaluate the trade-off offered by MAP-Me through the adjustment of this T_U parameter by slightly modifying our baseline scenario. Instead of a grid, the PoA are arranged on a line. The producer now moves back and forth across them at a constant speed parameter, while the consumer is now static at the root of the fat tree.

Fig. 5.9(a) shows the average path stretch of MAP-Me as function of Tu. The dashed line indicates the path stretch limit reached when no IU is sent. In general path stretch slowly increases with T_U at any given speed and remains well below the no-IU threshold.



(a) (b)

FIGURE 5.9: Effectiveness of Tu timer: a) Path stretch b) Network overhead (No. of updated routers per handover).

At low speed, stretch remains constant up to higher T_U values (as an IU is sent for every handover).

If we now consider network overhead depicted in Fig. 5.9(b), we notice that a slight increase of path stretch allows for a significant reduction of network overhead (which peaks here at 50% for a speed of 15m/s). This confirms the interest of notifications in absorbing high-frequency mobility while preserving appropriate flow performance. The T_U threshold thus appears a useful setting to allow a network to cope with challenging mobile workloads.

5.5.6 Trace-driven urban mobility

Topology: To evaluate our approach under more realistic mobility patterns, we consider an urban residential environment spanning a 2.1×2.1 km^2 area in Los Angeles, with a WiFi Hot Spot deployment similar to what Time Warner Cable [148] has in the area, i.e. we dislocate 729 WiFi APs, with the same wireless settings as in the previous (baseline) experiments, connected to the Internet through the fat-tree topology in Fig.5.5.

Mobility: We generate realistic vehicular mobility patterns using SUMO [83], with maximum car speed set according to road speed limits[4]. We place mobile producers in moving cars and analyze system dynamics on a given time interval (4 minutes, roughly corresponding to 33 handovers), so that all monitored cars are in the map at the same time. In such a scenario, we consider a group communication between one mobile producer and two non mobile consumers requesting different data. Consumers are connected to two APs that are picked at random, uniformly across the network coverage.

[4]In the selected area we have three different road categories characterized by different speed limits: 40, 70 and 55 km/h.

Applications: Two types of applications are considered: in the first set of simulations, the previously detailed streaming application is characterized by a rate of 1Mbit/s. In the second set, a pseudo real-time video streaming application, reproducing the popular application Periscope [122] is used. The mobile producer generates two different video streams, each one downloaded and played by one consumer, using a 5 s play-out delay buffer. If the video play-out stops because the consumer has no Data available, we consider this as a failure and momentarily stop the consumer: after a short period of time (few seconds), the consumer restarts downloading new data and to play-out the video. The video data rate is 1Mbit/sec, corresponding to a 480p video resolution. Traffic is scaled up by increasing the number of groups, identified by the producer serving data.

5.5.6.1 User Performance

To quantify user experience, we analyze the following metrics: the average packet loss and user satisfaction, while varying the number of mobile producers in the area (from 1 to 50, each one serving two consumers).

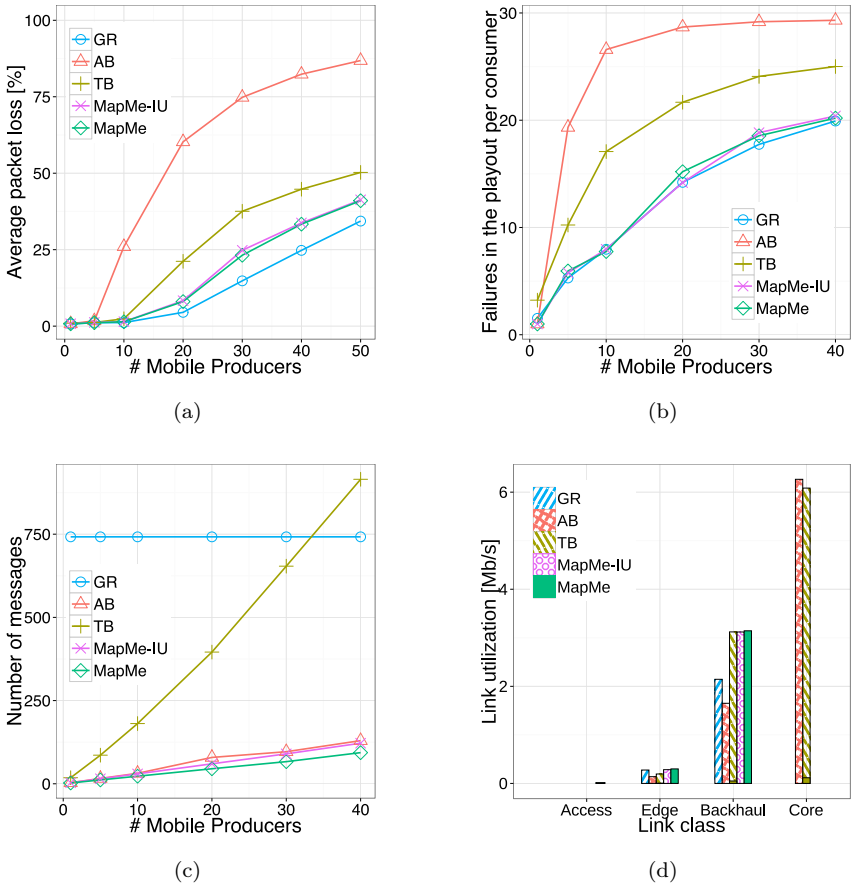


FIGURE 5.10: User performance: CBR average packet loss (a), Periscope playout failures (b). Network Cost: CBR overhead (c) and Periscope link utilization (d).

Packet loss: We evaluate the distribution of packet losses per second for the CBR application. Fig. 5.10(a) shows the average packet loss for MAP-Me and other protocols, while increasing the number of mobile producers in the system. As expected, increasing the number of active users in the network has a negative effect on the performance, because links are getting congested and routers start to lose packets. However, as shown in Fig. 5.10(a), the performance of MAP-Me and MAP-Me-IU is close to the ideal GR, while TB leads to higher loss rate and with AB, we observe an even more rapid increase in packet loss. Indeed, the distributed nature of MAP-Me allows the proposed solution to better cope with an increasing number of mobile producers.

User Satisfaction: We evaluate user satisfaction by analyzing the number of failures in the play-out of the video stream for the pseudo real-time video streaming (Periscope-like). Fig. 5.10(b) shows the number of failures in the video play-out that each consumer encounters in 4 min. As in the CBR case, when the number of mobile producers increases, the performance of the system degrades. Similar to what is observed in CBR case before, AB concentrates all traffic on a single node, the anchor, thus giving rise to congestion. In contrast, distributed protocols such as MAP-Me are able to better distribute traffic over the network and thus better cope with larger number of users. For the same reason, TB performs better than AB, but worse than MAP-Me/GR. Indeed, sending traces to the anchor forces traffic towards upper layers in the network, preventing substantial traffic offload at the edge. It is worth noticing that the application runs a classic window-based congestion control without video rate adaptation and no specific mobility loss recovery mechanisms (all packet losses are recovered based on timer expiration at the consumer). The design of such schemes and the analysis of the interaction with mobility-management protocols is out of the scope of this work and left for future work.

These simulations clearly show the effectiveness of MAP-Me in dealing with high loads as it spreads traffic over a more diverse set of paths.

5.5.6.2 Network Cost

Beyond user performance, we evaluate MAP-Me in terms of network cost, by computing the overhead and comparing it with all other considered solutions. Fig.5.10(c) reports the overhead computed as the number of messages exchanged in the network at each handoff, whereas Fig.5.10(d) displays link load distribution across the network (in the case of 10 mobile producers in the map). The figures prove that MAP-Me successfully offloads the core from local traffic with light overhead, in virtue of the absence of fixed anchor deep in the network.

5.5.6.3 Network topology and Mobility

Trace-based simulation have been run with pedestrian mobility and a tree-like network topology [21]. Results show the same behavior for vehicular and pedestrian mobility, while in the case of tree topology TB and MAP-Me have similar packet loss (due to higher chances of congestion at the core of the network).

5.6 MAP-Me and routing

While MAP-Me can efficiently manage producer mobility by updating FIB entries, it might however interfere with routing protocol as both can update FIB concurrently. In this section, we discuss their coexistence and show that minimal requirements on the routing and minor modification to MAP-Me can allow for both to perform correctly and asynchronously. We conclude by preliminary insights into their joint performance.

5.6.1 Proposed solution

Our proposal makes minimum assumptions on properties of the routing protocol: (i) the routing protocol is *link-state* — so that every node gets a sense of routing convergence state —; (ii) every router maintains a counter R_{seq}, incremented each time a non-duplicated routing message (LSA) is received – **Rseq** is expected to be either available or easily deducible from routing; and (iii) a routing instance is also running on the producer so that the producer is informed of network changes. We assume the router generating a new prefix advertisement or detecting a link failure will also increment this counter for global consistency.

On MAP-Me side, the idea is to delay MAP-Me's operation on a node until routing seems to converge locally (by checking $R_{\mathbf{seq}}$). We achieve this through a minor modification to the original design: upon sending a special interest, the sequence number field is augmented with the local $R_{\mathbf{seq}}$ information. When IU/IN is received, additional checks are performed before standard MAP-Me operation: by comparing R_{seq} in IU/IN ($R^{\text{IU}}_{\text{seq}}$) and the local one from routing ($R^{\mathbf{loc}}_{\mathbf{seq}}$). **Case (i)** if $R^{\text{IU}}_{\text{seq}} = R^{\mathbf{loc}}_{\mathbf{seq}}$, the producer and the nodes might be synchronized, and standard operations can proceed; **case (ii)** if $R^{\text{IU}}_{\text{seq}} > R^{\mathbf{loc}}_{\mathbf{seq}}$, the node has not received all routing updates and the IU is queued until $R^{\text{loc}}_{\text{seq}}$ gets incremented by routing, and eventually the IU pass through the node; **case (iii)** if $R^{\text{IU}}_{\text{seq}} < R^{\mathbf{loc}}_{\mathbf{seq}}$, the IU is discarded as all downstream nodes does not receive all routing updates.

Finally, to ensure correctness, we require the producer to issue a new IU each time it receives new routing messages (i.e, $R_{\mathbf{seq}}$ incremented). This IU corrects the route if

routing recomputes route towards producer's old location due to network changes and unawareness of producer's new location.

5.6.2 Correctness

This scheme ensures full producer reachability upon global convergence. Considering a single producer update during routing convergence, it is easy to see that the corresponding IU will traverse all routers that have seen the same number of routing updates as the producer. It is otherwise either delayed by case (ii) or dropped (iii). The last IU sent by the producer is guaranteed to complete (as there are no routers with higher R_{seq}, and that the forwarding tree is consistent as all routers have then the latest routing state.

During routing instabilities, there is no guarantee of connectivity and the forwarding state might not be loop-free either. It seems natural that we cannot require MAP-Me to improve on that situation. The design of a joint routing and mobility management protocol, following the same principle as MAP-Me, is an interesting direction left for future work.

5.6.3 Evaluation

We now illustrate the behavior of the modified algorithms, and analyze the effect of routing updates frequency on system performance. We consider the previous baseline scenario with 1 pair of mobile nodes, and a speed of 10m/s. The producer triggers a new routing update with varying frequency.

Routing convergence time obviously impact performance significantly. It is generally considered that link-state IGP convergence time is in the order of several seconds. while [51] demonstrates the possibility for sub-second convergence time for large ISP networks by leveraging techniques like fast flooding and incremental FIB updates, it is not widely deployed. We thus reasonably assume the routing convergence time lies in the order of sub-second to several seconds. In the evaluation we choose between 600ms and 6s.

Figure 5.11 illustrates the trade-off in setting the routing update frequency. Obviously, more frequent updates allow for shorter paths as they are re-optimized more often (Fig. 5.11(a)). However, instabilities due to routing at global scale lead to long-lasting unreachability of the producer after he moves, and thus a high packet loss rate (Fig 5.11(b)). Routing updates should thus be limited or triggered carefully, for instance in periods of producer stability (e.g based on mobility prediction).

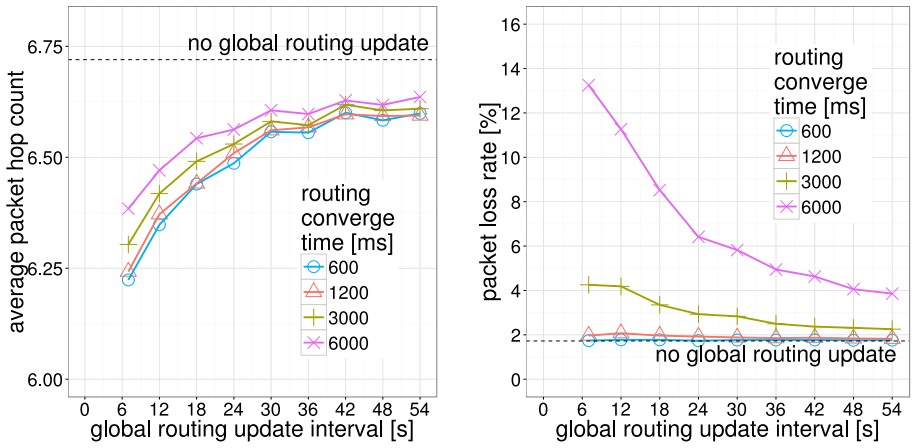


FIGURE 5.11: MAP-Me and routing. Effects of routing update frequency on performance: (a) Packet loss rate. (b) Path stretch.

5.7 Conclusions

Native support for mobility management at network layer is a recognized strength of NDN, and appears to be a key feature to exploit the design of 5G networks. However, a comprehensive solution for mobility management in NDN (and ICN in general) still lacks: previous attempts so far have either tried to apply Mobile IP concepts to ICN or looked at partial aspects of the problem, without providing a thorough evaluation of the initial solutions sketched in ICN context. The contribution of this work is twofold. First, we define MAP-Me, an NDN-based Local Routing approach for managing intra-AS producer mobility even in the presence of latency-sensitive traffic. By design, MAP-Me is simple as it only leverages NDN forwarding plane and reactive notifications to the network, is lightweight in terms of required signaling messages and, to our knowledge, the first one with proven guarantees of bounded stretch and overall correctness for the forwarding update process. Second, we open-sourced a simulation framework on top of NDNSim 2.1 using model-based and trace-driven consumer/producer mobility patterns over many topologies, and integrating anchor-based and tracing-based approaches, a reference implementation for MAP-Me, as well as a global routing approach useful for comparison. Evaluation takes 802.11n access in small cell outdoor settings and proves WiFi can support mobility using NDN in general settings.

The reported results confirm our initial objectives and show that MAP-Me optimally offloads the infrastructure from communications that are local. All other approaches making use of a fixed anchor, which in practice is also the network gateway, can be optimized only if traffic is non local. Instead, the current propositions in 3GPP to offload the mobile network core stem from the observation that, on the contrary, communications

are most likely local. On the other hand, MAP-Me would serve non-local communications through one or multiple gateways without binding mobility feature to any specific location.

Chapter 6

ParkMaster: visual analytics at the edge

6.1 Introduction

The previous chapters demonstrated how the concept of "connected car" can be achieved thanks to NDN. V-NDN, Navigo and MAP-Me showed that, even in a highly mobile environment as VANETs, connectivity between cars and to the Internet can be provided, and that vehicles are not only the end points of the communication, but also forwarding nodes and mobile caches. Starting from this point — cars are connected — we now extend the role of vehicles to mobile computing nodes and show, with a practical use case, the potential of having mobile computing resources with Internet connectivity (cars) spread all over a city.

Urban driving can be challenging and stressful, with the task of searching for parking spaces one of the key reasons. For example, a 2007 study in San Francisco [139] shows that in one of that city's commercial districts a driver spends on average 6.5 minutes to find a parking spot after reaching the destination, adding about 3.3 km to the trip. The same study shows that every day in one Los Angeles commercial district alone, cars searching for a parking space generate more than 3,500 vehicle-miles of travel (VMT)—more than a coast-to-coast US journey. Hence simply steering drivers to the closest available parking spot would address a significant portion of the traffic congestion and pollution present in big cities, with the added benefits for drivers of increased convenience and saved time.

This chapter presents the design and implementation of ParkMaster [58], a Fog-based computing system that leverages the ubiquitous smartphone to help drivers find parking spaces in the urban environment.

While there are several systems that let drivers see parking garage availability [142], providing such information for road-side parking spots is more challenging because they are often not clearly delineated with road markings. Recently Google announced that Google Maps started providing drivers with statistical information about how difficult parking is at the destination area [5], information that however is based on historical patterns and has a coarse granularity (the probability of finding a spot is deemed Easy, Medium or Limited). Focusing on real–time data instead, some cities, such as San Francisco [136], are currently installing sensors on parking meters to detect cars' presence. These solutions however entail significant infrastructure deployment, with an associated cost. Other prototype systems use image recognition techniques to detect cars' presence in parking lots by using cameras placed on top of buildings or poles [47, 161, 150]. While feasible for single parking lot, these systems suffer the same infrastructure costs. [113, 101, 141, 112] instead exploit user's smartphone in order to track the owner's actions (parking the car or leaving the spot). However, those system are unaware of other drivers (who may not use these systems) and thus require a high penetration rate to maintain a reasonable accuracy. [112] tries to overcome such a limitation by designing a non-user's actions prediction model to estimates the effects of nonusers to parking availability. However, such a model focuses on large parking lots and not on the more challenging on-street parking scenario. The ParkNet System [107] takes a complementary approach, adding ultrasonic sensors on cars to flag empty roadside parking bays. While ParkNet achieves high accuracy using mobile sensors, it again requires the installation of additional hardware on cars, posing deployment and cost issues. Nonetheless, the core idea of the vehicles themselves operating as a sensor network to collect data about parking availability remains promising. Such a collection system would indeed facilitate coverage over a large geographical area, without the burden associated with adding roadside infrastructure.

There are three key technology trends also changing the problem space:

1. Image processing has made significant accuracy advances in accuracy and speed.
2. Mobile devices' processing capacity and camera quality have both been steadily increasing over recent years, making now feasible to run more advanced image–based machine learning algorithm on smartphones.
3. Cloud services are being restructured to push latency-critical parts of their functionality out to the edge (Fog computing).

Viewed together, these three trends suggest a new design point: a system that leverages users' smartphones as sensors to capture parking information at the network's edge in real-time, reducing the cost for cities to only that required for the cloud service.

In this chapter, we demonstrate that such an edge-based sensor system is feasible with nowadays hardware: we present the design and implementation of ParkMaster, a system

FIGURE 6.1: ParkMaster deployed in a car's windshield.

that estimates parking space availability using video gleaned from drivers' dash-mounted smartphones on the network's edge, uploading analytics about the street to the cloud in real time as participants drive. Novel lightweight parked-car localization algorithms enable the system to estimate each parked car's approximate location by fusing information from phone's camera, GPS, and inertial sensors, tracking and counting parked cars as they move through the driving car's camera frame of view. To visually calibrate the system, ParkMaster relies only on the size of well-known objects in the urban environment for on-the-go calibration. We implement and deploy ParkMaster on Android smartphones, uploading parking analytics to the Azure cloud. While we believe that future cars will be equipped with computing power that could substitute smartphone's CPU in ParkMaster's computation, in this chapter we look at the phone in order to have a deployable system with nowadays vehicles. It must be noticed, however, that the problematics addressed by ParkMaster remains the same in the in-car computation scenario — computing resources are still limited and needs to be used with parsimony.

ParkMaster uses the cameras on drivers' phones to sample the presence of cars at road-side parking spots from the driver's vehicle itself. It features two main components. Firstly, the ParkMaster *app*, which runs on the in-car edge — the driver's smartphone — performs real-time visual analytics. Secondly, the ParkMaster *cloud service* maintains a real-time database summarizing the number of available parking spaces on each road and provides client support for location services. While the user is driving, with the smartphone placed on the windshield as shown in Figure 6.1, ParkMaster captures video with the phone's camera and, locally processing frames in real-time, looks at the road-side parking spaces.

The most natural approach to parking-space detection may seem to be searching for empty spaces, as humans do. Hence we have investigated edge detection [37] algorithms on the smartphone to detect and measure free parking spaces themselves. However, while feasible in controlled environments when the background is uniform, this approach quickly becomes more challenging in real cities where road and background scene have an arbitrary appearance. Furthermore, there are no guarantees that an empty stretch of

shoulder is a legal parking spot, and detailed maps with exact parking space coordinates (with meter–level accuracy) are unavailable in most locations. Also, smartphone absolute localization accuracy is too imprecise to reliably identify parking spots. We further contrast ParkMaster with this approach in Section 6.2.1.

We therefore take the approach where the smartphone detects the parked cars themselves. The smartphone then estimates the location of each detected vehicle, streaming this information to the cloud over Wi-Fi or the cellular interface. In order to detect the presence of a parked car at the roadside, ParkMaster needs to recognize and track the parked car as it moves across the frame of the driving car's camera. Existing motion tracking algorithms [43, 86, 156, 169] excel at this when the camera is still or almost still and objects are moving against background, but to the best of our knowledge, do not focus on our scenario where both background and object are moving across the frame together.

Having made the above design choice of detecting parked cars, in order to estimate free parking space, ParkMaster's cloud service relies on an additional information feed: the number of parking slots per road. This data is nowadays available in many cities, for example the cities of Seattle and Paris maintain this data online. At a high level, ParkMaster estimates the parking availability of each *road segment* (the smallest piece of street connecting two intersections) as the difference between its parking capacity and the number of parked vehicles that the cloud estimates by aggregating the parking analytic data that the smartphones upload. Finally, ParkMaster's cloud assists the smartphones in their own localization process by providing GPS data correction. Since the only hardware ParkMaster requires is users' off-the-shelf smartphones, ParkMaster benefits from the individual deployment of each user's smartphone.

Contributions.

1. We demonstrate the feasibility of running image–based machine learning techniques at the edge, on today's smartphones, to capture valuable information about the surrounding environment in real time without any human intervention, when battery power is not a concern. In order to do so we tackle the available parking location problem designing, building, and deploying a complete system.

2. We propose a novel, lightweight tracking algorithm for car detection that fuses speed estimates of the car with vision processing of the video stream to "de-duplicate" multiple detections of the same parked car in a drive, while the background scene moves with the parked car (unlike most object tracking algorithms).

3. We design and implement a localization algorithm to estimate the location of a parked car in a single frame, without requiring stereo vision or any input from the user, instead relying on camera calibration against well-known objects in the driving

environment. Our localization algorithm is lightweight and thus can efficiently run on a smartphone.

Roadmap. The rest of this chapter is organized as follows. Section 6.2 describes the design of ParkMaster, while Section 6.3 summarizes its implementation. A performance evaluation containing microbenchmarks and a real-world evaluation of ParkMaster on the streets of two major cities (Los Angeles and Paris) and an Italian village (Sant'Angelo in Vado) follows in Section 6.4. ParkMaster achieves a 90% average end-to-end accuracy, the result of meter-level video based localization of cars from the phone's camera. A sensitivity analysis of ParkMaster's design parameters explores optimal camera and video parameters to use given the smartphone's available processing power, and shows a tradeoff between false positive car detections and missed cars. Section 6.5 surveys related work, and we conclude in Section 6.6.

6.2 Design

This section describes the ParkMaster design in detail. Starting with functionality on the edge, we first explain how ParkMaster recognizes (§6.2.1) and localizes (§6.2.2) vehicles parked on the road. Continuing with functionality located in the cloud, we then describe how ParkMaster counts parked cars and free spots on a road (§6.2.3). We begin with the main goals of our design, followed by a high-level design summary to place each part into context.

Design goals. ParkMaster has the following goals:

1. Since computation at the edge is limited, we avoid overly-expensive computation on the mobile.
2. Since real-world conditions and camera limitations may obscure line of vision or create visual ambiguities, we aim to design a system that can identify a car given a limited number of frames in which the car is visible.
3. Operating within the constraints of the first design goal, since smartphone-cloud data constitutes a cost for the users, we design a system that leverages edge processing to limit the amount of data uploaded.

Non goals. Phone power management: Here we assume that user powers the smartphone from the car's electrical system.

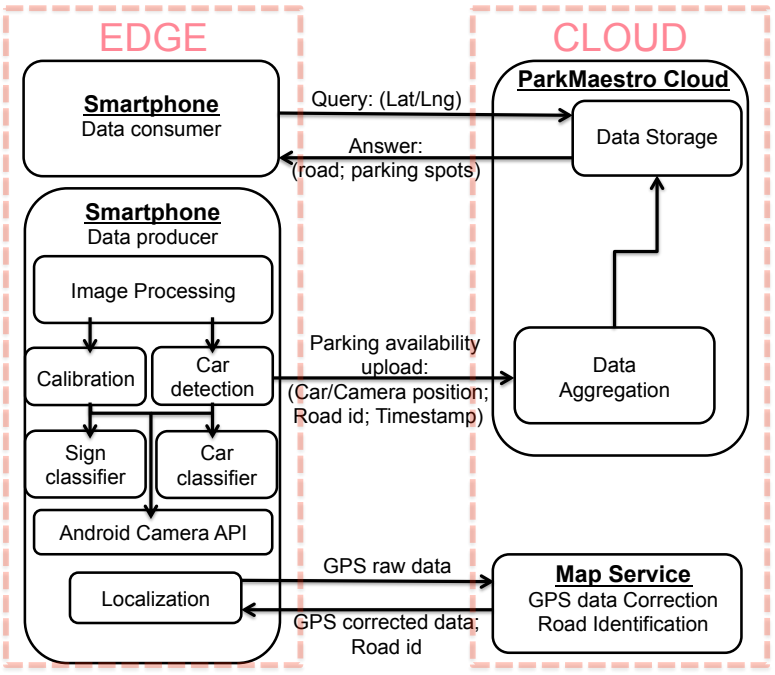


FIGURE 6.2: ParkMaster architecture.

Design summary. We present ParkMaster's overall architecture in Figure 6.2. Edge-side software running on the smartphone begins with the *camera calibration* phase, processing the video stream looking for road signs. For each frame containing a road sign of known height and size, ParkMaster computes its real-world coordinates (relative to the camera) and records the relationship between these relative real-world coordinates and the sign's coordinates in the smartphone's two-dimensional camera field of view. As soon as ParkMaster collects enough samples, it runs the calibration algorithm described in Section 6.2.2, to derive a rotation and translation vector that captures the phone's three-dimensional orientation. Once calibration concludes, ParkMaster loads a car classifier and starts searching for cars parked on the right side of the road, parallel to the street (Section 6.2.1). On detecting a parked car, ParkMaster extracts its frame coordinates, computes the parked car's real-world coordinates (relative to the camera) and stores both sets of coordinates and the time of detection on the phone: we refer to these data together as the *parked car analytics* (Section 6.2.2). Simultaneously, ParkMaster records data from the phone's GPS, accelerometer, magnetometer and gyroscope sensors, leveraging Google Maps' "snap to road" functionality [57] to obtain even better phone localization accuracy. Whenever the user enters a new road ParkMaster retrieves the samples collected from the previous road segment and computes the camera's position (in real-world coordinates) at each detection time. This estimate, together with the relative real-world coordinates of each parked car, yields latitude-longitude coordinates of the vehicles. If the sample is deemed valid, ParkMaster uploads the parked car analytics stored at the edge to the cloud, adding the car's real-world latitude and longitude, a road identifier, and the time each sample was detected. When the cloud receives a phone's update it runs the DBSCAN [46] clustering algorithm in order to count the number of

parked cars on the road segment. Finally, ParkMaster subtracts this estimate from the number of parking spots to compute the number of free parking spaces.

In the future, as new data arrives about road conditions, the cloud discards old data in favor of more recent information provided by other users. When a driver queries the ParkMaster cloud looking for a parking spot, the latter provides the computed per road-segment parking availability of the area, together with its timestamp, which allows drivers to evaluate the freshness of ParkMaster's information.

In contrast with [113, 101, 141, 112], ParkMaster collects data about the surrounding parked cars — not about users actions — thus it doesn't need to run in every vehicles nor to make additional efforts in guessing non-users' behavior.

It must be noticed, furthermore, that by only uploading the parked car analytics instead of the full video stream, the privacy of "third-persons" in the area is preserved: ParkMaster doesn't store or move across the network any frame or information about pedestrian or cars (i.e. no plate, model, color etc.). The only information exchanged between edge and cloud is the estimated coordinates of anonymous parked cars.

6.2.1 Detecting parked cars

Works like [82] have shown the feasibility of running image–based basic techniques like color filtering and edge detection on smartphones to detect particular events while driving, like traffic light status. ParkMaster takes a step further and applies machine learning techniques, the Viola-Jones feature-based cascade classifier [160, 95], to detect complex objects — cars — in the phone camera's video stream, in real time.

During *one-time offline training*, the classifier learns about the object's features through sets of positive and negative examples of the object of interest. Once the training terminates, the classifier's *online detection* (on the smartphone) analyzes the video stream looking for the features listed by the classifier in a multi-stage process. For each object that reaches the final stage, the classifier provides the *frame coordinates* of a bounding box containing the object, as shown in Figure 6.3.

To search for larger or smaller examples of the object of interest, at each stage the features of interest are scaled, based on a *scaling* parameter. To reduce the number of spurious object detections, a candidate part of the image is considered detected only if there are at least k adjacent detections in its immediate vicinity (*i.e.*, one to two pixels): k is referred to as the *minimum neighbors threshold*.

ParkMaster must operate in a variety of light levels, *i.e.* sunny days *v.* cloudy days; time of the day *i.e.* mornings, afternoons, evenings; and direction of the light. Ideally ParkMaster could use a specialized classifier and/or different detection settings for each situation. However, to reduce system complexity, we chose to use a single classifier

FIGURE 6.3: Parked car detection: frame samples indicating bounding box of detected cars.

trained in for a wide range of conditions. As shown in Section 6.4, we have obtained good results running ParkMaster in three different countries across a wide variety of ambient conditions.

Discussion. In order to find empty parking spots the most natural approach is to look for empty spaces. However, such approach has two limitations: first, a map with the coordinates of each parking spot (with meter–level accuracy) is required to determine if the empty stretch of shoulder is a legal parking spot, information that is rarely available. Second, smartphone localization accuracy doesn't allow to properly identify a spot (a few meters error in the localization may translate in referring to the wrong stretch of shoulder in a hypothetical parking coordinates database). In contrast ParkMaster relies in a coarser and easier to get knowledge, the number of parking spots per road, which is already available online for instance for cities like Seattle and Paris[1], and doesn't need a precise object localization; instead it simply requires that estimated locations of neighboring cars don't completely overlap — they don't need to match any specific location in the map.

6.2.2 Localizing parked cars

It is likely that the same vehicle is detected multiple times in subsequent frames. Thus, to accurately count the number of parked cars, ParkMaster has to determine when a vehicle has been already detected in previous frames, and count it only once. Analyzing subsequent frames, for instance computing pixel–based image difference, to track a vehicle, is highly CPU intensive and doesn't easily handle differences among subsequent detections of the same car (while the driver is approaching the parked vehicle, the view of the car and the background changes). Thus, with its primary design goals in mind, ParkMaster aims to estimate each detected cars' latitude-longitude position in the real world, and then decide whether two detected cars are in fact the same car based on their computed coordinates.

As described in the previous section, the detection algorithm only provides the position of the parked car in the coordinates of the camera's frame. In the remainder, we refer to

[1] See web6.seattle.gov/sdot/seattleparkingmap/ or opendata.paris.fr.

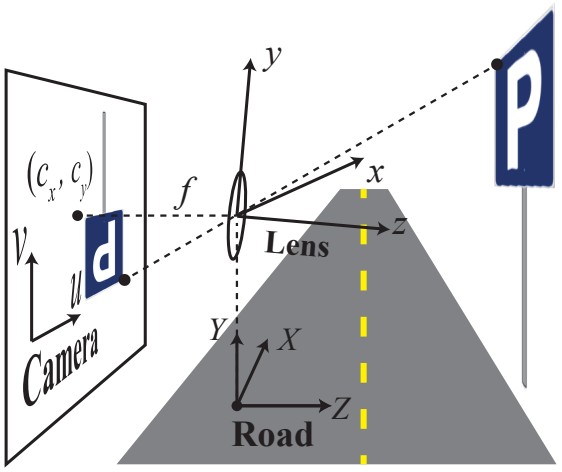


FIGURE 6.4: Coordinate systems used in ParkMaster's design: the calibration process maps objects in the **camera** coordinate system to the real-world **road** coordinate system through the **lens** coordinate system.

a car's position in the camera's two-dimensional coordinate system (u, v) with *camera* coordinates (u_0, v_0). In order to compute the corresponding latitude-longitude in the real world ParkMaster must map from camera coordinates to a three-dimensional *road* coordinate system (X, Y, Z), whose X-Z plane is parallel with the road and whose origin is located on the road under the camera as shown in Figure 6.4. This is the classic *camera calibration* problem, which we now describe.

6.2.2.1 Camera calibration

Camera calibration uses the classical *pinhole camera model*, which treats the camera's aperture as a point in space. Although this model does not account for various factors such as lens distortion, coordinate discretization, and blurring, it does provide a good first-order approximation to the camera's projection. Camera calibration amounts to estimate two types of parameters. The first type, *intrinsic camera parameters*, characterizes unchanging optical properties of the camera: *focal length* f, the distance between the pinhole and the film, and offsets of the axes from the origin in the film c_x and c_y. The second type, *extrinsic parameters*, describes camera's location in the world via a translation vector $\mathbf{t}$ and the direction in which the camera points via a *rotation matrix* $\mathbf{R}$. In contrast with the intrinsic parameters, the extrinsic parameters $\mathbf{R}$ and $\mathbf{t}$ change as the camera points in different directions.

In order to separate the effects of the intrinsic and extrinsic parameters, we introduce a new coordinate system (x, y, z), dubbed *lens* coordinate system, whose z-axis is aligned with the camera lens. Now the mapping from lens to camera coordinate systems captures the intrinsic camera properties, while the mapping from road to lens coordinate systems captures all extrinsic camera properties (camera tilt, rotation, and elevation) except the

GPS coordinates of the camera: these we process as part of smartphone localization (§6.2.2.3).

Intrinsic parameter calibration. To find the focal length f and image center (c_x, c_y) parameters simply queries the Android camera API. Then, it can translate between the lens and camera coordinate systems with the following relationships of the pinhole camera model:

$$x = (u - c_x) \cdot z/f \qquad \text{and} \qquad y = (v - c_y) \cdot z/f. \tag{6.1}$$

Extrinsic parameter calibration. To find the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$, ParkMaster calibrates the camera while driving—without user input—using objects of well-known sizes that can be easily found on the road. In both the United States and Europe, road signs have a consistent height $h_{\text{sign}}^{\text{road}}$ (measured in road coordinates) and elevation from the road $Y_{\text{sign}}^{\text{road}}$ mandated by the highway code.

ParkMaster uses a separate cascade classifier trained on a sign of interest to search for road signs during a "start-up phase". Every time the sign classifier detects a sign, it returns the camera coordinates of the detected sign (u_0, v_0), as well as the height of the detected sign in camera coordinates $h_{\text{sign}}^{\text{camera}}$. These, along with the intrinsic camera parameters described above, allow us to compute the camera's range to the sign Z_0 as a function of the sign's height in camera coordinates:

$$Z_0 = h_{\text{sign}}^{\text{road}} \cdot f / h_{\text{sign}}^{\text{camera}} \tag{6.2}$$

To avoid using the results of calibration before the calibration itself is over, we approximate $X_0 \approx x_0$, and take $Y_0 = Y_{\text{sign}}^{\text{road}}$. This gives us a pair of (camera, road) coordinates for the single sign detection. After L detections, the list

$$\left[\left((u_0, v_0), (X_0, Y_0, Z_0)\right), \ldots, \ \left((u_{L-1}, v_{L-1}), (X_{L-1}, Y_{L-1}, Z_{L-1})\right)\right] \tag{6.3}$$

of (camera, road) coordinate pairs can be passed to OpenCV [118], a computer vision library, which calculates the extrinsic parameters $\mathbf{R}$ and $\mathbf{t}$ based on a global Levenberg-Marquardt optimization algorithm that minimizes projection error [33, 182].

Notice that in order to estimate Z_0 and X_0, we have assumed that the X and Y axes of the road and the lens coordinate systems are aligned. Not having this requirements satisfied, which is most likely to happen in a realistic scenario (as drawn in Figure 6.4), introduces error in the calibration. The misalignment is unlikely to be large in realistic driving settings as the driver wants to see the phone's display, and thus can be tolerate. In order to verify this empirically, we intentionally placed the smartphone not always perfectly aligned with the road axis in our experimental evaluation (§6.4.1).

Once $\mathbf{R}$ and $\mathbf{t}$ are computed, they are valid till the smartphone remains in its original position. Currently ParkMaster assumes the driver won't remove the phone from its holder while driving. However, if this happens, it is straightforward to pause the car detection, reset $\mathbf{R}$ and $\mathbf{t}$ and start again the calibration phase once the phone is back in place (this event could be triggered manually by the driver or automatically by the phone's sensors). Minor changes in the phone's position, due for instance to car vibration, have instead a negligible and usually temporary effect on $\mathbf{R}$ and $\mathbf{t}$ (the phone holder stabilizes the device) and thus ParkMaster doesn't try to compensate for such events. In the experimental evaluation (§6.4.1) we used a standard phone holder without any additional actions to improve phone's stability, thus the results reported include the effects of vibrations, potholes etc.

6.2.2.2 Car localization

After calibration, ParkMaster is ready to localize parked cars. However, camera parameters do not suffice to find a unique position in road coordinates given a set of camera coordinates, since a coordinate in the camera frame corresponds to multiple points in the road frame. In order to find a unique road coordinate, we exploit a natural constraint of our scenario: cars elevation from the ground is always zero, *i.e.*, $Y = 0$. Thus, supposing car detection has just detected a parked car at camera coordinates (u_c, v_c), our task is to determine X_c and Z_c, the two dimensional location of the car in the road coordinate system. To this end, consider the relationship between the road and lens coordinate systems:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{R}^{-1} \cdot \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \mathbf{t} \right). \tag{6.4}$$

With Equations in 6.1 we can compute (x_c, y_c), the (x, y) position of the car in the lens coordinate system. Subsituting in $(x \leftarrow x_c, y \leftarrow y_c, Y \leftarrow 0)$ and the calibrated values of $\mathbf{R}$ and $\mathbf{t}$ into Equation 6.4 yields a linear system of three equations in three unknowns (X, Z, z), which has a unique solution (X_c, Z_c) for the car's road-coordinate location.

6.2.2.3 Smartphone localization

Now that we have a mean of mapping car locations between camera and road coordinate systems, for each frame where a car is detected, ParkMaster extracts the lower-middle point of the bounding box (Figure 6.5) and computes its road coordinates. This measure, together with the coordinates of the smartphone, allows ParkMaster to estimate the absolute position of the detected vehicle and compare it with the coordinates of subsequently-detected cars.

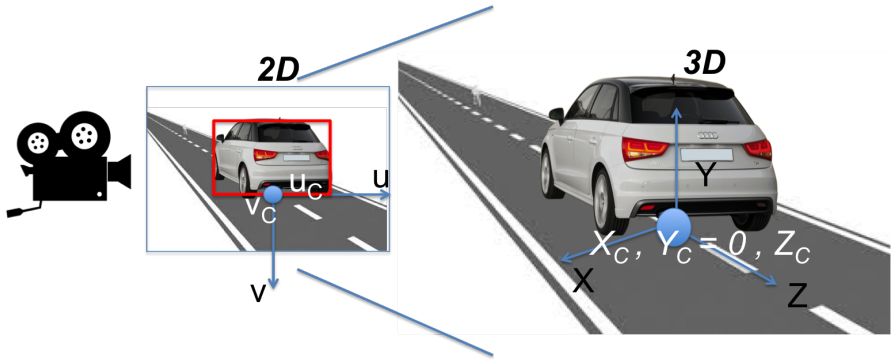


FIGURE 6.5: Extracting the lower-middle point of a detected car's bounding box in the camera coordinate system for processing in the road coordinate system, to yield a final GPS location estimate of the parked car.

Nowadays, every smartphone is equipped with a GPS sensor. To improve its accuracy, ParkMaster periodically sends raw GPS data to Google Maps ("snap-to-road") API service, which supports filtering and correction of the GPS traces: it removes or corrects points out of the road, interpolates coordinates based on street layout and identifies the road segment on which the user is driving. This information flows into the next stage of ParkMaster's processing, counting parked cars.

6.2.3 Counting parked cars

Now that it has an approximate GPS location for each parked car detection, ParkMaster relies on car's approximated coordinates to track its location between successive frames and estimate the number of parked cars. Since the localization process is prone to error, we can't do a mere coordinate comparison, because (1) the detection algorithm doesn't always pick the same point of a car; (2) the localization process is affected by inaccuracy, due to calibration imperfection and camera distortion; and (3) the smartphone location given by GPS and snap-to-road is not always accurate.

Most likely, whenever ParkMaster detects a car multiple times, it gets a set of coordinates close to each other (a couple of meters). Thus, in order to distinguish among different cars, ParkMaster runs a Density Based Spatial Clustering of Applications with Noise (DBSCAN[46]) algorithm on all the coordinates collected on a segment of road, clustering nearby points into estimates of a single car's location. Knowing the maximum number of cars that a given road can accommodate and the number of estimated cars on that road, ParkMaster estimates the parking availability.

Preliminary studies have been done in order to choose the proper clustering algorithm: among the approaches which don't require prior knowledge of the total number of clusters, we evaluated Affinity Propagation [53], Mean Shift [42] and DBSCAN. In the counting cars problem these techniques seem equivalent, with DBSCAN that slightly outperforms the others in CPU time. It must be noticed, however, that due to the small

number of samples (few points per car), in most of the cases the difference in performance is negligible.

Discussion. The estimate of the maximum number of parking spaces a road can host may affect the accuracy of ParkMaster. Indeed, while sometimes road-side parking is marked with bays, in others they are left unmarked, making the maximum number of parked vehicles on the road merely an estimate. At the moment ParkMaster does not take any additional action to cope with the uncertainty of unslotted areas, but we are confident that with more sophisticated techniques (*i.e.*, measuring space between parked cars) we will be able to improve accuracy in free parking scenarios.

6.2.3.1 Heuristics

In order to increase accuracy ParkMaster applies the following heuristics during the clustering phase:

Sample distance. Increasing the camera-vehicle distance generally increases the error of the parked car's localization, which may lead to unreliable estimation (§6.4.4). Thus ParkMaster discards samples estimated to lie further than a certain threshold distance, determined empirically in §6.4.1.1.

Single-element cluster. Today's smartphones can't process every video frames in real-time (§6.4.1.3). Furthermore, cars are not always detected, even when clearly visible. As a result, some vehicles are detected only once. To decrease the number of misses, ParkMaster conservatively considers elements that don't belong to any DBSCAN cluster as individual parked vehicles. We note here that while this approach will rarely result in spurious parked vehicle detections, our experimental evaluation shows that it has an overall benefit to accuracy.

Cluster size. Consecutive cars may be merged into the same cluster if the points in the overall dataset are dispersed and the clustering process fails to discriminate between them. In this case ParkMaster relies on the "size" of the cluster to evaluate the number of vehicles. If the maximum distance $d_{\max}$ among two points in the same cluster is bigger than a certain threshold *maxClusterSize*, ParkMaster splits the cluster into n smaller clusters, where $n = d_{\max}/maxClusterSize$.

Driving cars. Even though the detection algorithm is trained with pictures of vehicles capturing the back and part of the side of a car, it may happen that the classifier detects vehicles that are driving in the opposite direction (only the front is visible) or that are

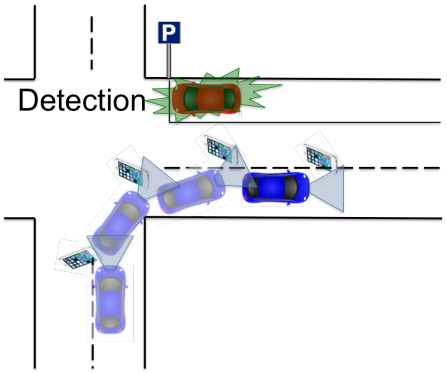


FIGURE 6.6: Car parked on the opposite side of the street.

driving in front of the camera (only the back is visible). To mitigate these cases, the classifier focuses only on the lower-right part of the video, where cars parked on the right side of the road usually appear. Reducing the area of interest also allows the classifier to process only a portion of the frame, which reduces the per-frame processing time.

A special case is represented by multi-lane roads, where vehicles on the right may be moving cars. In this case, car detection must be enabled only when the user is driving on the rightmost lane. While at the moment ParkMaster assumes the user is always doing so, we believe vision-based [125] or sensors-based [31, 134, 14, 15, 40] solutions for lane detection can be easily integrated with ParkMaster to trigger the detection only when the user is driving on the right lane.

Samples while turning. A vehicle parked on the left side of the road may fell in the lower-right part of the video, resulting in false detections (Figure 6.6). Since GPS sometimes fails to discern these cases, ParkMaster also samples phone's magnetometer, accelerometer and gyroscope, and, when approaching an intersection, discards the sample if the user is turning i.e. the current orientation of the phone differs from the average orientation the phone had on that road.

6.3 Implementation

Edge. We have implemented ParkMaster on three mid-range to high-end Android phones: a Samsung Galaxy S4 GT-19505 running Android 4.4.2, a Samsung Galaxy S6 edge+ running Android 5.1.1, and an LG Nexus 5 with Android 6.0.1. The Galaxy S4 and the Nexus 5 are equipped with a Quad Core CPU and 2 Gbytes RAM, while the Galaxy S6 is equipped with two Quad Core CPUs and 4 Gbytes RAM. All smartphones are equipped with at least an eight megapixel main camera as per the "full" specifications [1, 2, 3].

For cascade classifier-based car detection on the phone, we use OpenCV 2.4, an open-source library for image processing available for Android. The classifier has been trained

with 1,202 positive samples and 719 negative samples, resulting in 20 stages. Currently, ParkMaster focuses on parallel parking only, thus we train the classifier using pictures capturing the back and the side of the vehicle as positive examples.[2]

Cloud. We have implemented ParkMaster cloud services in Azure, using Azure Mobile Service for cloud backend functionality and Azure mobile app client and Node.js SDK for authentication and interaction between clients and the in-cloud database (MongoDB) used for data collection. For GPS data correction and road identification instead the Google Snap Road API has been used. The API is available on Android as part of the Java Client library for Google Maps.

6.4 Evaluation

In this section we present a comprehensive performance evaluation of ParkMaster. We begin with single-driver, real-world experiments that exercise ParkMaster entire processing pipeline in "on the road" scenarios in metropolitan and rural environment (§6.4.1).We then discuss data coverage and freshness (§6.4.2). Finally, we drill down into ParkMaster design, explaining how we have tuned parameters in the car detection algorithm (§6.4.3) and car localization algorithm (§6.4.4).

6.4.1 Road-based experiments

We experiment in both metropolitan (Los Angeles and Paris) and rural environments (Sant'Angelo in Vado, a small village in Italy). In Paris cars are typically parked 30–40 cm. from each other, while in Los Angeles they are usually separated by larger space and in the European village they are typically spaced by a half-meter or more, and sometimes isolated (*i.e.*, a few solitary parking spots on a long road). In addition, we experiment in different weather conditions: our tests span a period of ten days between December 2015 and March 2016, between 11 a.m. and 6 p.m. Typical weather includes a clear sky, dawn, partly cloudy conditions, and cloudy with light rain. In total, as Table 6.1 shows, we drove for 97 km. on roads that have at least one parking spot, covering a 30.5 km. path. We report a total of 5,896 parked cars and 2,280 available parking spaces[3]. Figure 6.7(a) shows these roads in a map view all three environments.

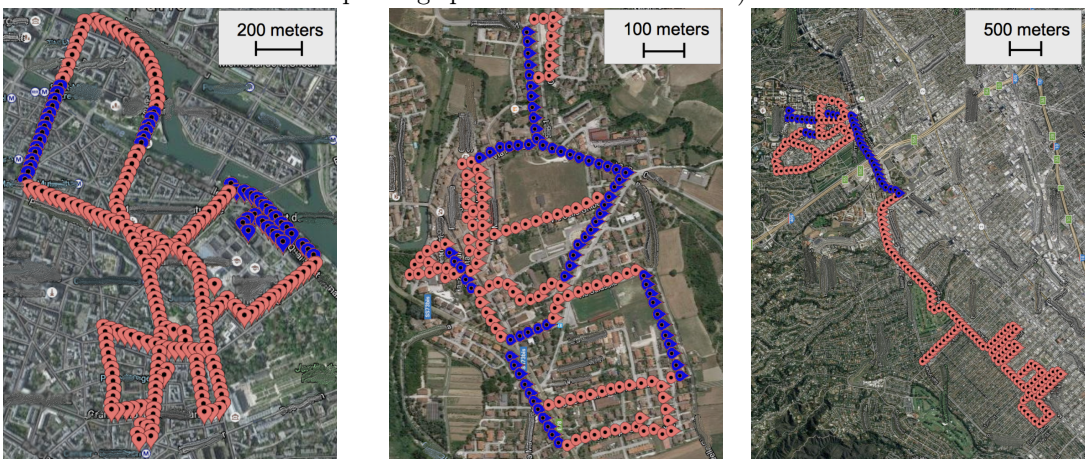
We experiment using the three phones described in Section 6.3 and drive three different cars (with different dashboard heights from the ground). In Paris we used the Galaxy S4

[2]It is possible to build a classifier for each type of parking (*i.e.*, angled or head-in parking) and then, knowing the type of parking spot on a given road, use the appropriate classifier.

[3]The amount of parking slots each road–segment can host has been computed manually, by on-the-field observations.

Place	Distance	Unique dist.	Slots	Parked cars
EU-village	17.9 km.	3.3 km.	1,381	710 cars
EU-city	38.4	7.9	3,527	2,892
USA-city	41.0	19.3	3,268	2,294
Totals	97.3	30.5	8,176	5,896

TABLE 6.1: Experiments on the road – covered distance (only roads with at least one parking spot have been accounted).



(a) EU-city. Paris (France). (b) EU-village. Sant'Angelo In Vado (Italy). (c) US-city. Los Angeles (USA).

FIGURE 6.7: Experiments on the road. Red points indicate streets with parking spaces, blue points indicate roads without legal parking.

and S6 respectively for 41% and 59% of the time, while we used the Nexus 5 for all in-village experiments and we equally used the Galaxy S6 and Nexus 5 in Los Angeles.

On each run, we place the smartphone as shown in Figure 6.1, slightly changing its position (i.e. few tens of cm) and orientation each time (i.e. few degrees). We run the calibration phase on a traffic sign, and then begin driving (on the rightmost lane), respecting local speed limits: 30 kph (Paris), 30 kph and 50 kph (Sant'Angelo in Vado), 40 kph and 48 kph (Los Angeles). On average, during each run we drive for 30 minutes. GPS sampling frequency is one sample per second.

6.4.1.1 Car detection accuracy

Before looking at ParkMaster entire pipeline, we begin with the detection phase in isolation, analyzing the effects of the heuristics described in Section 6.2.3.1. We empirically set the threshold of the above heuristics after preliminary experiments: we set the maximum sample distance to 7 m. (ParkMaster discards samples at larger distances) and the maximum cluster size to 6 m. (ParkMaster splits larger clusters into smaller groups). These values maximize detection accuracy which, however, was largely insensitive to their exact settings, varying less than 10% with a ± 1 m. difference.

ParkMaster	Ground truth: **Car**	Ground truth: **No car**
Car	Correct detection (TPR)	Spurious detection (FPR)
No car	Missed car (FNR)	Correct non-detection (TNR)

TABLE 6.2: Confusion matrix for accuracy evaluation.

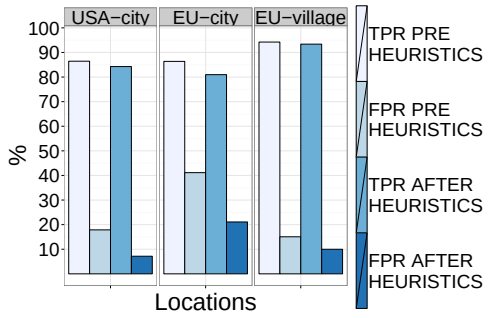
Accuracy metrics. In order to ascertain ground truth (the number of parked cars and empty parking spots) an additional camera records each experiment and we visually tally these quantities after the drive. We evaluate in terms of the *true positive rate* (TPR) and *false positive rate* (FPR), as shown in Table 6.2. TPR is the ratio of the number of cars that ParkMaster detects in at least one frame to the ground-truth number of parked cars, while FPR is the ratio of the number of spuriously-detected cars ParkMaster reports to the sum of spurious detection and correct empty parking space detection. If the same car is detected multiple times, we count it once. Note that whenever ParkMaster detects a car which is not parked or which is parked on the left side of the road, we count the sample as a spurious detection (false positive).

Results. Figure 6.8(a) shows the results. The heuristics of §6.2.3.1 slightly decrease the true positive rate. Indeed they sometimes fail discarding samples that are correct. In contrast they are strongly beneficial for the false positive rate. Manual data inspection shows that the heuristics are able to discard a good fraction of (1) running cars (based on road coordinates); (2) cars parked on the other side of the road when the user is turning (based on GPS and azimuth); (3) cars crossing intersections (based on GPS) and (4) non-vehicle object misclassified as cars that don't lay on the ground (based on their higher estimated distance to the camera).

6.4.1.2 End-to-end accuracy

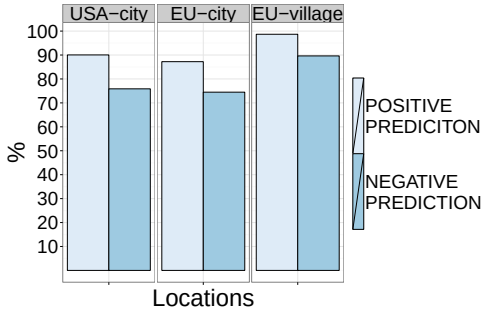
We now measure end-to-end accuracy, from detection on the smartphone to clustering and counting in the cloud.

Metrics. In order to evaluate end-to-end, we extend the above concepts in order to take into account the counting process. We consider true positive when a car parked on the right side of the road appears among the vehicles ParkMaster counted, while we consider as false positive all the non-cars, vehicles parked on the left side of the road and driving cars that ParkMaster counts as parked cars. Furthermore, whenever ParkMaster counts the same parked vehicle multiple times, we consider the first sample as true positive and all the others as false positives. In addition, we introduce the *accuracy with compensation* metric: the final outcome of ParkMaster is an estimation of the number of parked cars, which includes both correct samples and counting error—spurious detections



(a) **Detection** on the road: True and False Positive rates before and after applying the §6.2.3.1 detection heuristics.

(b) ParkMaster **end-to-end accuracy** in real-world driving experiments on the road.



(c) Can we park there? **Positive** (there is at least one spot) and **negative** (road is full) **prediction accuracy**.

FIGURE 6.8: On the road experiments

"compensate" misses in the count for a specific road. Thus, for each road i, we compute $error_i$ as the difference among the number of parked cars (ground-truth) and the number of vehicles ParkMaster considers as parked. We define the *accuracy with compensation* as:

$$accuracy = 1 - \frac{\sum_{i=0}^{\#roads} |error_i|}{\#\ parked\ cars}. \tag{6.5}$$

During a single experiment, it may happen that we drive several times on the same road; nevertheless, we consider each pass independent from the others—the error from one pass is not mitigated by later pass on the same road.

Finally, we evaluate ParkMaster accuracy when determining if there is space to park on a road in terms of *Positive* and *Negative Predictive Value*. Defining positive samples as occurrences of roads with at least one available parking spot and negative samples as occurrences of roads without free space, the *Positive Predictive Value* is the ratio of correct positive estimation (ParkMaster correctly estimates there is enough space) to the number of positive samples. Similarly, *Negative Predictive Value* is the ratio of correct negative estimations (ParkMaster correctly estimates the road is full) to the total number of negative samples.

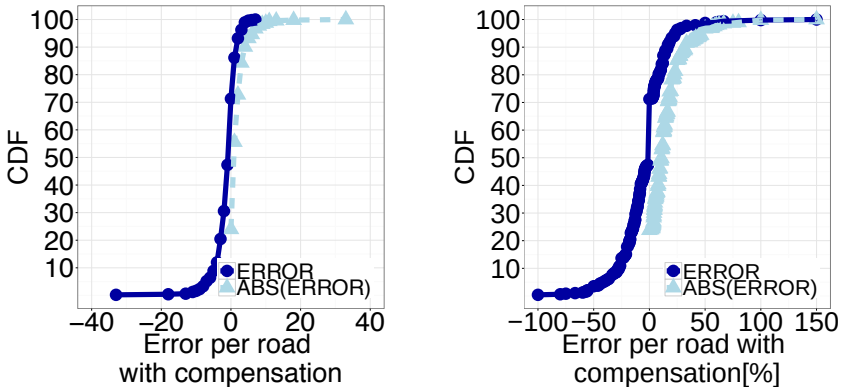


FIGURE 6.9: ParkMaster end-to-end counting accuracy in real-world driving experiments on the road.

Results. Figure 6.8(b) shows true and false positive rates and accuracy with compensation. In general, in rural environments ParkMaster shows better performance compared to urban scenarios. Indeed, in the first we usually have cars parked at larger distance, which overcomes eventual inaccuracies in the car localization estimates. In contrast, in Paris spaces among vehicles are limited and the counting process can tolerate a smaller error in the localization. Intermediate performance characterizes the experiments in USA, which indeed presents a higher density of cars than rural environments but a less chaotic parking displacement than the European city. Nevertheless, with its almost free of costs approach, ParkMaster always shows a satisfactory close to 90% accuracy (notice that the city of San Francisco considers their sensors-on-parking-meter deployment effective and starts paying for the service when accuracy is higher than 70% [6]).

Figure 6.8(c) shows Positive and Negative Prediction values. Similarly to Figure 6.8(b), rural environments show the highest accuracy. Anyhow in most of the cases (from 87% to 98%) ParkMaster successfully classifies roads with empty parking spots. Lower accuracy is reported for negative prediction. Indeed when a street is full or almost full, a single error may lead to a misclassification of the road status.

Figure 6.9 shows the effect of the error compensation on the accuracy of car counts. In particular, Figure 6.9(a) shows the error per road while Figure 6.9(b) shows the same error in percentage points (on the number of parking spots on the road). In both cases, the error is reported as absolute value and with a sign (a negative error means missing cars, while positive values mean ParkMaster overestimates the number of parked cars).

Error analysis. Inaccuracies can be caused by several factors. In particular, we define the following type of errors:

1. **Classifier:** the classifier detects something that does not correspond to a car,

	Classifier	B. B.	GPS	Loc.	F. P.
EU-city	24.3%	29.2%	26.3%	8.0%	12.2%
EU-village	29.6	29.8	14.4	7.0	19.2
US-city	24.5	27.8	25.1	17.1	5.6

TABLE 6.3: On-road experiments: Error analysis.

2. **B. B.:** Inaccurate bounding box—the classifier captures only the upper part of the vehicle,

3. **GPS:** ParkMaster fails to properly count cars due to GPS inaccuracies i.e. phone's GPS doesn't report any movement while car is moving (or the opposite), or GPS reports large distance among two consecutive samples (5-10 meters) while it's clear from the recorded video that user's movement is considerably smaller (1 meter or less),

4. **Loc.:** ParkMaster miscounts vehicles because the clustering process is not able to associate samples with the corresponding car i.e. a car has been counted more than once or multiple cars are merged into the same cluster,

5. **F. P.:** ParkMaster has failed to discard driving vehicles or cars parked on the left side of the road (false positive).

Table 6.3 shows the frequency of those errors during the experiments on–the–road. With the help of logs and videos we manually evaluate each erroneous sample and pick the best fitting error category. Table 6.3 confirms the remarkable impact that inaccuracy of bounding box and phone's GPS have on the car counting process. In particular, between the three scenarios, it shows a major impact of GPS error in the two cities. The difference is mainly due to higher buildings elevation (especially in the European case) and a more frequent *stop-and-go* car mobility caused by traffic lights and stops which characterize urban scenarios [38].

6.4.1.3 Processing rate

On todays phones ParkMaster doesn't keep up with video speed but drops frames: on average during the experiments Samsung Galaxy S4, S6 and Nexus 5 processed respectively 5.75, 10,3 and 6.75 frames per second. Despite the large gap between the two Galaxy phones, if we compare ParkMaster performance with the two phones in similar conditions (we run few experiments with both phones at the same time), the gap reduces: in those experiments the TPRs are 83.4% in the Galaxy S6 and 76.8% in the S4, while the FPRs are 34.4% and 27.8% respectively.

6.4.1.4 Data usage

By design, ParkMaster concentrates most of the computation at the edge and relies on the cloud only for data aggregation. The amount of information exchanged by the two parts is therefore negligible. During all the on-road experiments (about 8 hours driving), the phones exchanged 6.1 megabytes[4] with cloud and Google's snap-to-road service, divided as follows: (1) 67 bytes for each sample uploaded by the phone; (2) 16 bytes for each GPS coordinate sent to Google's snap-to-road service; (3) 47 bytes for each GPS coordinate corrected by Google's snap-to-road Service.

6.4.1.5 Fusion of parked cars analytics

One of ParkMaster's strength resides in being able to run in COTS smartphones, which potentially enables every driver to participate in data collection. As a consequence, the cloud may receives information about the same road from multiple drivers. While an extended study of multi–user data fusion is out of the scope of this work, we evaluate through preliminary experiments how an extended version of ParkMaster may reduce the error when users sample the same road in a short period of time (ParkMaster's outcome is likely to vary each time because of different user's speed, camera's properties, phone's characteristics etc.). In particular, we look into the single-element cluster case (object detected only once), which constitutes a considerable portion of false positives.

In this extended version of ParkMaster, the cloud, after receiving new data, instead of overriding older samples, assigns weights to each parked car detection, indicating its confidence. Whenever data from different users matches the same spot, samples are merged and their resulting weight is increased. In order to take into consideration time, the cloud periodically decreases these weights. Whenever a single-element cluster is uploaded, the cloud accepts it only if other users have recently reported a car at the same location — only if the resulting weight is higher than a certain threshold. To cope with GPS inaccuracy, the cloud may adjust traces of different users in order to match the position of the parked cars analytics.

In order to give a preliminary assessment of this approach, we place two smartphones (Samsung Galaxy S4 and S6) in the same car and we repetitively drive along the same path (950 m. path for a total of 11.4 km. in the Paris) to emulate multiple-users' activities in a short period of time.

Although such a data fusion approach is quite rudimental, it shows a reduction of 9.1% of false positives, with a small price in terms of true positive rate reduction (1.8%), which would further increase the accuracy of ParkMaster reported in Figure 6.8 (which have been obtained without data fusion).

[4]The negligible HTTP/TCP overhead has not been accounted.

We leave for future works a more extensive study of data fusion. For instance, external information like traffic reports i.e. Waze, or historical parking data, might be used to evaluate parked-cars-analytics validity over time.

6.4.2 Data pertinence to the parking search

As for any crowd-based system, the number of users collecting data is a crucial factor for the success of ParkMaster. Data coverage and freshness are indeed vitals to make ParkMaster information of any pertinence to the drivers looking for parking and strongly depend on the number of data collectors. In order to evaluate this, we utilizes results reported by [107]. Such a work, as previously mentioned, addresses the parking problem by installing additional hardware on cars. While the mean is different, the data collection model is similar: running cars collect data about parking availability in the area.

[107] analyzes taxis' traces collected over a month in the city of San Francisco [146] and estimates that with the only 536 cabs reported in the traces, in the downtown area of San Francisco, 80% of the road (dubbed cells in [107]) is on average visited with an inter-visit interval of under 10 minutes. Such a result shows that with a small subset of vehicles collecting data a significant data freshness can be achieved.

Moreover, it provides insights on how ParkMaster could be initially deployed: the system could run on taxis or other public vehicles, which has been shown to guarantee adequate data freshness, and then, given the low cost for the users (in contrast with [107]), expand to other drivers, for instance with the incentive of more parking queries for non-free-loaders.

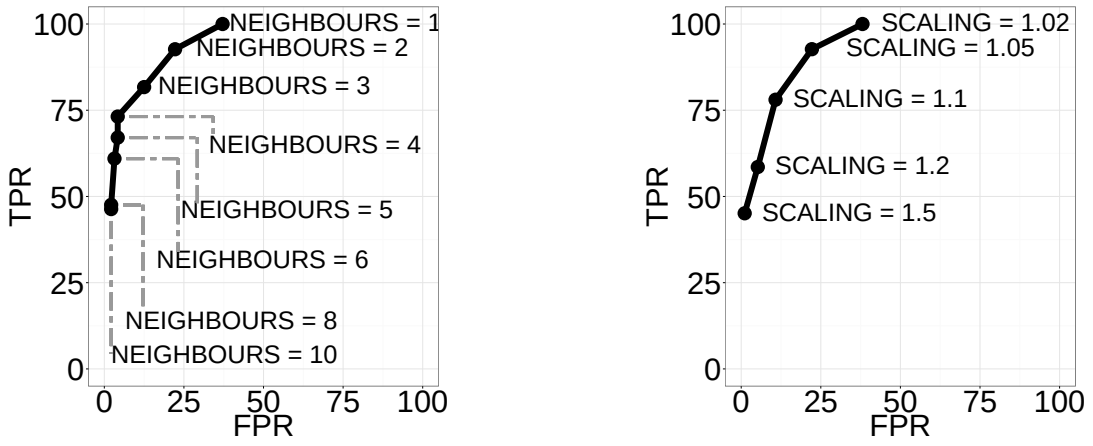
6.4.3 Tuning car detection

We now drill down into ParkMaster design, explaining how we have tuned parameters in the car detection algorithm.

We record a five-minute video using a phone while driving. Afterwards, we run the detection algorithm on a server, at first processing every frame of the video (30 fps), and then artificially skipping frames to emulate the edge limitations.

6.4.3.1 Tuning video parameters for accuracy

We first measure car-detection true and false positive rates at 30 fps while varying input resolution and the classifier parameters.



(a) *neighbors* threshold (*scale* = 1.05).

(b) *scale* parameter (*neighbors* = 2).

FIGURE 6.10: Tuning parameters of a computationally-unconstrained car classifier (1920 × 1080 resolution).

Viola-Jones classifier parameters. Figure 6.10(a) shows the impact of varying the *neighbors* threshold with a 1920 × 1080 resolution video and a *scale* = 1.05 (*i.e.*, scaling in steps of 5%) when the Jones-Viola classifier processes the video at full 30 fps rate. The ROC curve shows the tradeoff between fewer spurious car detections but more missed cars with a high *neighbors* (high detection confidence) and more detections (true and false) with low *neighbors*. Based on this data, ParkMaster sets *neighors* = 2.

Figure 6.10(b) shows the impact of varying the *scaling* parameter with the same resolution video and *neighbors* = 2. We see a similar tradeoff, with the classifier missing cars when its scaling search step is coarse (*scaling* = 1.5). Based on this data, ParkMaster sets *scaling* = 1.05 to make a reasonable tradeoff between misses and spurious detections.

Video resolution. Decreasing the video resolution reduces the FPR, but, at the same time, leads to lower TPR. While with resolutions higher than 720 × 480 the variation in TPR is relatively small, decreasing further the resolution causes a more consistent drop in the number of TP. Thus ParkMaster runs at 720 × 480.

6.4.3.2 Tuning video parameters for performance

Computational power at the edge is increasing, but still limited. We thus consider what performance level (as measured in frames per second the phone processes) we require. Figure 6.11 shows many missed cars at 2 fps and diminishing gains past 10–15 fps, because the classifier already has enough frames in which it may detect the car.

Table 6.4 summarizes a sensitivity analysis on performance, measured in frames per second the Samsung Galaxy S4 and S6 can process when only car detection is active.

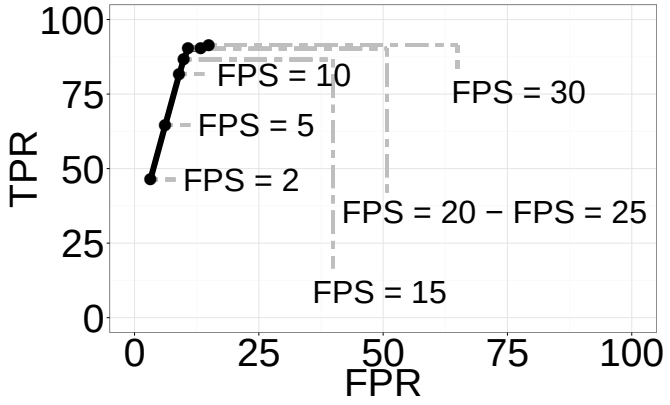


FIGURE 6.11: Detection with smartphone limitations: Impact of processing rate with video resolution 720 × 480, classifier parameters *scale* = 1.05 and *neighbors* = 2.

		Cropping		Processing rate	
Resolution	**Scale**	↕	↔	**S4**	**S6**
720 × 480	1.05	½	½	5.78 fps	10.4 fps ★
720 × 480	1.05	full	full	1.18	4.11
720 × 480	1.02	½	½	2.30	5.22
1280 × 720	1.05	½	½	2.08	5.90
1920 × 1080	1.05	½	½	0.90	3.45

TABLE 6.4: Sensitivity analysis of video parameters on phone performance (processing rate) measured in frames per second. ★ indicates the parameters ParkMaster uses.

Since ParkMaster aims to detect only vehicles parked on the right side of the road, it can focus on a smaller part of the image, the bottom-right part of the frame (½ cropping in both horizontal and vertical directions). We see that *scale* factors smaller than 1.05 significantly reduce the frames per second the smartphone can process. The data shows that with parameters chosen for ParkMaster (denoted by the ★ symbol), the Samsung S6 can process 10.4 fps, *i.e.*, one frame every 40 cm at 15 kph or one frame every 1.3 m. at 50 kph, removing frame rate as a limiting factor for car detection.

6.4.4 Measuring car localization error

In this section we probe the causes of inaccuracy in the localization process. Firstly, we evaluate the effects of camera distortion and calibration: we place an easily-recognizable object (a box with a high contrast color and well defined edges) at the end of a corridor, calibrating the smartphone with a copy of a road sign. Then, we place the phone at well-known positions in the corridor and we let it compute the relative position of the object to the camera. Due to the simplicity of the object, bounding box inaccuracies are close to zero.

Afterwards we measure the effects of bounding box misplacements — a classifier recognizing only parts of an object. We run a second, similar experiment in a parking garage detecting and localizing a real car.

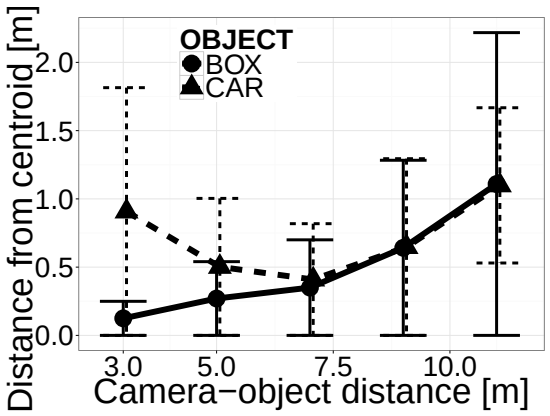


FIGURE 6.12: Localization error versus ground-truth distance from the object of interest, for an easily-recognized object indoors, and real cars in a parking lot.

In both experiments as the camera changes range to the object of interest, we compute the estimated location of the object (as camera-object distance plus camera position). Afterwards we compute the centroid of the set of samples collected at different locations and we measure the average distance between each sampled coordinate and the centroid: this essentially measures dispersion in the location estimate, which we refer to below as *localization error*. If the points are too dispersed, ParkMaster won't be able to track a car among subsequent frames because points belonging to cars close to each other will overlap.

Figure 6.12 shows the resulting localization error for both experiments. As expected, getting farther from the box increases the localization error, because a miscalculation in the estimation of the camera extrinsic parameters (i.e. wrong orientation) increases its effects at large distance. Similarly for the car experiment, which however presents a peak in the error at small distance, due to bounding box error: the classifier detected only part of the vehicle (*e.g.* only the upper part of the car), which translates in a further away estimated location in street coordinates.

6.5 Related Work

Car detection. *Advanced Driver Assistance Systems* use computer vision, among other techniques, to alert drivers of potential dangers in the surrounding area [144]. Several mechanisms have been developed to detect and track cars from a moving vehicle using an in-vehicle camera [27, 116, 63]. ParkMaster uses Viola and Jones classifier for its simplicity and computational speed.

Object tracking. Object tracking methods can be used in order to count cars. In [43, 86, 156, 169] the authors use background subtraction techniques to facilitate object tracking. This approach is however impractical in ParkMaster scenario, where the camera

is moving. [22, 43, 85, 92, 140] apply multi-object tracking-by-detection algorithms, which rely on large temporal video windows to discern among subsequent objects. In contrast in our scenario the same vehicle may be detected in few frames only. Finally [68, 133] combine particle filters with object detector for Markovian tracking-by-detection, which [34] uses to track pedestrians. The tracker initialization however requires the same object to be detected few times in subsequent frames, which again is not always feasible in ParkMaster scenario. Furthermore, the processing time of [34] is not negligible.

Object/obstacle localization. Stereo vision is commonly used to detect obstacles and estimate their relative location in the scene using simple triangulation [168, 114, 149]. Stereo vision comes at the cost of two cameras facing a scene at the same time, not available in the majority of COTS smartphone. In [105] authors use different pictures of the same scene taken by one camera in different locations to emulate two virtual cameras and reconstruct a 3D model of the scene. This approach however is very CPU intensive. Similarly Wedel *et al.* [167] propose the use of multiple pictures taken by a single camera to estimate the distance of a vehicle from a moving car. In order to be accurate the algorithm needs several samples, which may not be always available in ParkMaster case. Furthermore, both [105, 167] require an accurate localization of the camera, which may be hard to achieve with smartphone's sensors while moving on a car.

Camera calibration and coordinate system mapping. Another class of single-camera based methods for object localization consists in mapping camera's two-dimensional coordinate system to the three-dimensional road coordinate system. [151] introduces a pattern-based camera calibration, which has been used by [69] for obstacle avoidance on smartphones (through chessboard camera calibration). However, to calibrate the phone, the driver has to stand in front on the car with a chessboard every time the phone is placed on the dashboard. In contrast in ParkMaster, given the triggering action — the phone has been placed — calibration runs on–the–go without user intervention. [152, 126, 73] exploit Inverse Perspective Mapping (IPM) to remove perspective effects. Such techniques however require to know the height of the camera and it's angle to the road [104], which may change at each run. Similar assumptions are made by [174], which needs camera's height to compute car-camera distance.

6.6 Conclusion

This chapter has described ParkMaster, the first system to combine advanced vision algorithms, Fog computing on mobile devices, and the cloud to build a zero-overhead parking space availability solution for dense, vibrant urban environments that costs drivers and the city next to nothing. On–the–road experiments run in uncontrolled environments

(city of Paris, Los Angeles and a small Italian village) validate ParkMaster approach, whose accuracy in estimating parking availability reaches 90%. We believe ParkMaster and similar technologies will push the envelope of what is possible in the smart cities of tomorrow, with a key-role played by the edge in extracting vital information about the urban environment.

Chapter 7

Conclusion

In the recent years the concept of smart city — a fully integration between urban environment and technology — has gain popularity. The city of the future will be invaded by IoT devices and sensors connected to the Internet which will participate in our lives and in every urban activities. Among those smart and connected devices, cars will play a fundamental role. With their networking, computation and sensing capabilities indeed, newly manufactured vehicles can become data provider, consumer, forwarder and processor.

In-car sensors indeed can produce data by reporting the vehicle's status or by sensing the environment. This information, as well as other data produced by IoT devices deployed around the city or data published on the Internet, can be consumed by other cars and passed by from one vehicle to another through multi-hop communication. Finally, before being shared or utilized, data produced in-loco by cars can be processed (at least partially) by in-loco smart devices instead of relying only on remote servers (i.e. the cloud), limiting the dependency to external and remote entities.

Even though the connected car in the urban environment is a well-known concept, it still presents unresolved challenges, mainly due to the highly mobile nature of vehicles, which makes paths more unreliable and the data more troublesome to reach, and the difficulties in exploiting at the same time the diverse network interfaces the car may be equipped with. In the literature plenty of works based on IP have been proposed over the year, but still no final solution has been found. Indeed, we believe the main problem resides in the IP design, which focuses on the endpoints of the communication instead of its content, making the data exchange susceptible to any change in location or of network interfaces of any of the two endpoints. We thus believe that to achieve the final solution to the connected car problem, a shift in the network model is needed, towards a paradigm that focuses on the content of the communication rather than its endpoints. Therefore, we adopt the Named Data Networking architecture and with **V-NDN** (Chapter 3) we demonstrate, for the first time, the feasibility, benefits and challenges of such approach

in urban vehicular networks with real experimentations on the field. With V-NDN, the first implementation of NDN for VANET running on real hardware, we propose some adaptations of the NDN paradigm to cope with the challenges of vehicular networks (e.g. mobility and multihoming) and to exploit their potentials (e.g. caching capabilities, broadcast nature of the communication, mobility i.e. moving data around without any transmission). V-NDN demonstrates that (1) NDN supports multihoming by design, by decoupling the communication from its endpoints, from the network interfaces in use; (2) naming data and caching enable the content to survive the producer — once the data is out and has been cached by some cars in the network, others can retrieve it simply by name. There is no need anymore for the producer, the actors of the exchange may not have been part of the initial communication and it's not required for data mules to know anything about the semantic of the data name and the nature of the carried information. V-NDN's goal was not only demonstrating the benefits of NDN, but also highlighting its unresolved challenges. V-NDN indeed shows that, while NDN paradigm is beneficial for the disruptive nature of the vehicular networks, it's not ready to a real deployment at scale. By on-the-field experiments and simulations, V-NDN shows that (1) to obtain scalability a way to steer Interest towards the data without flooding the network is needed; (2) similarly, a smart forwarding strategy able to select the best network interface to use is necessary, (3) NDN still misses a transparent way to exploit the locality of some of the VANET typical data — information produced in a specific area (i.e. traffic information) are likely to be found in that area — which can bring great benefits to the communication performance; (4) finally, security and data validation (i.e. can the information produced by a car be trusted?) are still an open research problem.

Navigo (Chapter 4) addresses some of those challenges and proposes a location based packet forwarding mechanism for V-NDN. Navigo takes a radically new approach to address the challenges of frequent connectivity disruptions and sudden network changes in a vehicle network. Instead of forwarding packets to a specific moving car, Navigo aims to fetch specific pieces of content from multiple potential carriers of the data. It automatically learns content's geographical location, without the need of any location service or oracle which are typically required by traditional Geo-routing. The design provides (1) a mechanism to bind NDN data names to the producers' geographic area(s): Navigo allows the consumer to register location-based names with the area where such a content can be found and provides automatic mechanism to create this binding when the consumer is unaware of the data location; (2) it provides an algorithm to guide Interests towards data producers using a specialized shortest path over the road topology; and (3) provides an adaptive discovery and selection mechanism that can identify the best data source across multiple geographic areas, as well as quickly react to changes in the V2X network. Navigo has been extensively evaluated through simulations and features low overhead and high performances for both V2V and V2I scenarios. Those studies also confirmed that NDN's basic breadcrumbs mechanism is resilient to mobility: the 95^{th} RTT percentile for an Interest-Data transaction is less than 300ms, which ensures the

validity of the Interest breadcrumbs in the PIT — vehicles do not move far in the time elapsed between an Interest and the corresponding data.

NDN design choice of decoupling the communication from its endpoints and addresses and of focusing on the information exchanged instead, makes it the perfect candidate to support mobility management at network layer and be a key factor to exploit the design of 5G networks, thus not only for vehicular networks, but in general for any mobile connected device. However, while consumer mobility is natively supported by NDN (a consumer simply needs to issue again an Interest in case of handover), the producer mobility is still an open problem. With **MAP–Me** (Chapter 5) we present an NDN-based model for managing intra-AS producer mobility events in the presence of latency-sensitive traffic. MAP-Me leverages NDN forwarding plane and notifications to the network after a handoff to make data reachable after each producer's movement, while remaining lightweight in terms of required signaling messages. This work presents a theoretical evaluation of MAP-Me, proving the guarantees of bounded stretch and overall correctness for the forwarding update process, and evaluates MAP-Me performance by simulation. The results show that, whenever the communications are local, MAP-Me optimally offloads the infrastructure from the traffic, in contrast with other anchor-based approaches, which concentrates all the traffic towards a single node in the network. When instead traffic in not local, the results show that MAP-Me is still able to serve the communications without binding mobility feature to any specific location, while preserving the communication performance.

The role of cars in the smart city of the future doesn't stop to the networking aspect. Once the "connected car" concept is achieved, indeed, new opportunities (and challenges) arrive at the horizon. The in-car computation and sensing capabilities and the large diffusion of vehicles all across the urban environment make the car the perfect candidate to play the role of edge node in the Fog architecture. With **ParkMaster** (Chapter 6) we demonstrates that this is not just a vision for the future, but that the current technology and the current diffusion of smart devices already enables vehicles to sense the local environment, locally process the collected data and make the resulting view available to the rest of the world. In particular, with ParkMaster we present the first system that, by combining vision and machine learning algorithms, the Fog computing model on mobile devices and the cloud, enables car to provide a zero-overhead parking space availability solution for urban environments that costs drivers and the city next to nothing. ParkMaster uses cameras of drivers' smartphones to capture images of the road and to process the pictures looking for parked vehicles, using a Viola-Jones classifier. The resulting information about parked cars locations is then uploaded to the cloud, which aggregates the data and makes it available to other drivers looking for a parking spot. To cope with the computational limitation of smartphones, ParkMaster introduces a novel lightweight tracking algorithm for car detection based on the location estimates of each sample in order to "de-duplicate" multiple detections of the same vehicle. Finally, it proposes a new

localization algorithm to estimate parked car location with a single frame. On–the–road experiments run in uncontrolled environments (city of Paris, Los Angeles and a small Italian village) have shown that ParkMaster's accuracy in estimating parking availability reaches 90%, which demonstrates that vision-based sensing of the environment at the edge is now possible with the current technology.

Bibliography

[1] http://www.gsmarena.com/samsung_i9505_galaxy_s4-5371.php.

[2] http://www.gsmarena.com/samsung_galaxy_s6_edge+-7467.php.

[3] http://www.gsmarena.com/lg_nexus_5-5705.php.

[4] DOT proposes mandatory V2V. https://www.transportation.gov/briefing-room/us-dot-advances-deployment-connected-vehicle-technology-prevent-hundreds-thousands.

[5] Google maps now makes it easier to find parking. https://techcrunch.com/2017/08/29/google-maps-now-makes-it-easier-to-find-parking/.

[6] Parking sensor performance standards and measurement. http://sfpark.org/wp-content/uploads/2011/09/SFpark_SensorPerformance_v01.pdf.

[7] A. Afanasyev, I. Moiseenko, L. Zhang. ndnSIM: NDN simulator for NS-3. Tech. Rep. NDN-0005, NDN Project, 2012.

[8] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. DiBenedetto, G. Grassi, *et al.* Nfd developer's guide. Tech. rep., Technical Report NDN-0021, NDN, 2014.

[9] A. Afanasyev, C. Yi, L. Wang, B. Zhang, L. Zhang. SNAMP: Secure namespace mapping to scale NDN forwarding. 281–286, 2015.

[10] B. Ahlgren, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, V. Vercellone. Design considerations for a network of information. 66, 2008.

[11] S. H. Ahmed, S. H. Bouk, D. Kim. Rufs: Robust forwarder selection in vehicular content-centric networks. *IEEE Communications Letters*, **19**(9), 1616–1619, 2015.

[12] V. Albino, U. Berardi, R. M. Dangelico. Smart cities: Definitions, dimensions, performance, and initiatives. *Journal of Urban Technology*, **22**(1), 3–21, 2015.

[13] M. Ali. *Green Cloud on the Horizon*, 451–459. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-10665-1. doi:10.1007/978-3-642-10665-1_41.

[14] H. Aly, A. Basalamah, M. Youssef. Lanequest: An accurate and energy-efficient lane detection system. *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*, 163–171. IEEE, 2015.

[15] H. Aly, A. Basalamah, M. Youssef. Robust and ubiquitous smartphone-based lane detection. *Pervasive and Mobile Computing*, **26**, 35–56, 2016.

[16] M. Amadeo, C. Campolo, A. Molinaro. Content-centric networking: is that a solution for upcoming vehicular networks? *Proceedings of the ninth ACM international workshop on Vehicular inter-networking, systems, and applications*, 99–102. ACM, 2012.

[17] M. Amadeo, C. Campolo, A. Molinaro. Crown: Content-centric networking in vehicular ad hoc networks. *IEEE Communications Letters*, **16**(9), 1380–1383, 2012.

[18] M. Amadeo, C. Campolo, A. Molinaro. Design and analysis of a transport-level solution for content-centric vanets. *Communications Workshops (ICC), 2013 IEEE International Conference on*, 532–537. IEEE, 2013.

[19] M. Amadeo, C. Campolo, A. Molinaro. Enhancing content-centric networking for vehicular environments. *Computer Networks*, **57**(16), 3222–3234, 2013.

[20] G. Arnould, D. Khadraoui, Z. Habbas. A self-organizing content centric network model for hybrid vehicular ad-hoc networks. *Proceedings of the first ACM international symposium on Design and analysis of intelligent vehicular networks and applications*, 15–22. ACM, 2011.

[21] J. Augé, G. Carofiglio, G. Grassi, L. Muscariello, G. Pau, X. Zeng. MAP-Me: Managing Anchor-less Producer Mobility in Content-Centric Networks. Technical report, `https://mapme-tnsm17.github.io/`, 2016.

[22] S. Avidan. Ensemble tracking. *IEEE TPAMI*, **29**(2), 261–271, 2007.

[23] A. Bachir, A. Benslimane. A multicast protocol in ad hoc networks inter-vehicle geocast. *Vehicular Technology Conference, 2003. VTC 2003-Spring. The 57th IEEE Semiannual*, vol. 4, 2456–2460. IEEE, 2003.

[24] P. Bahl, R. Y. Han, L. E. Li, M. Satyanarayanan. Advancing the state of mobile cloud computing. *Proceedings of the third ACM workshop on Mobile cloud computing and services*, 21–28. ACM, 2012.

[25] F. Bai, B. Krishnamachari. Exploiting the wisdom of the crowd: localized, distributed information-centric vanets [topics in automotive networking]. *IEEE Communications Magazine*, **48**(5), 2010.

[26] T. Bakıcı, E. Almirall, J. Wareham. A smart city initiative: the case of barcelona. *Journal of the Knowledge Economy*, **4**(2), 135–148, 2013.

[27] M. Betke, E. Haritaoglu, L. S. Davis. Multiple vehicle detection and tracking in hard real-time. *Proc. of IEEE Intelligent Vehicles Symposium*, 351–356, 1996.

[28] C. Bian, T. Zhao, X. Li, W. Yan. Boosting named data networking for data dissemination in urban vanet scenarios. *Vehicular Communications*, **2**(4), 195–207, 2015.

[29] C. Bian, T. Zhao, X. Li, W. Yan. Boosting named data networking for efficient packet forwarding in urban vanet scenarios. *Local and Metropolitan Area Networks (LANMAN), 2015 IEEE International Workshop on*, 1–6. IEEE, 2015.

[30] J. Blum, A. Eskandarian, L. Hoffman. Mobility management in ivc networks. *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, 150–155. IEEE, 2003.

[31] C. Bo, X.-Y. Li, T. Jung, X. Mao, Y. Tao, L. Yao. Smartloc: Push the limit of the inertial sensor based metropolitan localization using smartphone. *Proceedings of the 19th annual international conference on Mobile computing & networking*, 195–198. ACM, 2013.

[32] F. Bonomi, R. Milito, J. Zhu, S. Addepalli. Fog computing and its role in the internet of things. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 13–16. ACM, 2012.

[33] J. Bouguet. MATLAB calibration tool: `http~://www.vision.caltech.edu/bouguetj/calib_doc/`.

[34] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, L. Van Gool. Robust tracking-by-detection using a detector confidence particle filter. *IEEE Conf. on Computer Vision*, 1515–1522, 2009.

[35] L. Briesemeister, L. Schafers, G. Hommel. Disseminating messages among highly mobile hosts based on inter-vehicle communication. *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, 522–527. IEEE, 2000.

[36] A. T. Campbell, J. Gomez, S. Kim, A. G. Valkó, C.-Y. Wan, Z. R. Turányi. Design, implementation, and evaluation of cellular ip. *IEEE personal communications*, **7**(4), 42–49, 2000.

[37] J. Canny. A computational approach to edge detection. *IEEE TPAMI*, (6), 679–698, 1986.

[38] R. Carisi, E. Giordano, G. Pau, M. Gerla. Enhancing in vehicle digital maps via gps crowdsourcing. *WONS*, 27–34. IEEE, 2011.

[39] G. Carofiglio, L. Muscariello, ichele Papalini, N. Rozhnova, X. Zeng. Leveraging icn in-network control for loss detection and recovery in wireless mobile networks. *ACM SIGCOMM ICN'2016*. Kyoto, Japan, 2016.

[40] D. Chen, K.-T. Cho, S. Han, Z. Jin, K. G. Shin. Invisible sensing of vehicle steering with smartphones. *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, 1–13. ACM, 2015.

[41] P.-C. Cheng, K. C. Lee, M. Gerla, J. Härri. Geodtn+ nav: geographic dtn routing with navigator prediction for urban vehicular environments. *Mobile Networks and Applications*, **15**(1), 61–82, 2010.

[42] D. Comaniciu, P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE TPAMI*, **24**(5), 603–619, 2002.

[43] N. Dalal, B. Triggs. Histograms of oriented gradients for human detection. *IEEE conf. CVPR*, vol. 1, 886–893, 2005.

[44] C. Dannewitz, D. Kutscher, B. Ohlman, S. Farrell, B. Ahlgren, H. Karl. Network of information (netinf)–an information-centric networking architecture. *Computer Communications*, **36**(7), 721–735, 2013.

[45] M. Durresi, A. Durresi, L. Barolli. Emergency broadcast protocol for inter-vehicle communications. *Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on*, vol. 2, 402–406. IEEE, 2005.

[46] M. Ester, H. Kriegel, J. Sander, X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, 1996.

[47] T. Fabian. An algorithm for parking lot occupation detection. *CISIM*, 165–170. IEEE, 2008.

[48] B. Feng, H. Zhou, Q. Xu. Mobility support in named data networking: a survey. *EURASIP Journal on Wireless Communications and Networking*, **2016**(1), 220, 2016.

[49] Strategic Opportunity Analysis of the Global Smart City Market. 2013. Frost and Sullivan.

[50] N. Fotiou, P. Nikander, D. Trossen, G. C. Polyzos, *et al.* Developing information networking further: From psirp to pursuit. *Broadnets*, 1–13, 2010.

[51] P. Francois, C. Filsfils, J. Evans, O. Bonaventure. Achieving sub-second igp convergence in large ip networks. *ACM SIGCOMM Computer Communication Review*, **35**(3), 35–44, 2005.

[52] R. Frank, E. Giordano, P. Cataldi, M. Gerla. Trafroute: A different approach to routing in vehicular networks. *Wireless and Mobile Computing, Networking and Communications (WiMob), 2010 IEEE 6th International Conference on*, 521–528. IEEE, 2010.

[53] B. J. Frey, D. Dueck. Clustering by passing messages between data points. *Science*, **315**(5814), 972–976, 2007.

[54] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, E. Riviere. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, **45**(5), 37–42, 2015.

[55] E. Giordano, R. Frank, G. Pau, M. Gerla. Corner: a realistic urban propagation model for vanet. *Wireless On-demand Network Systems and Services (WONS), 2010 Seventh International Conference on*, 57–60. IEEE, 2010.

[56] V. Giruka, M. Singhal. Hello protocols for ad-hoc networks: overhead and accuracy tradeoffs. *World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a*, 354–361, 2005. doi:10.1109/WOWMOM.2005.50.

[57] Google snap to road. `https://developers.google.com/maps/documentation/roads/snap`.

[58] G. Grassi, P. Bahl, K. Jamieson, G. Pau. Parkmaster: An in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments. *Symposium on Edge Computing (SEC), 2017*. ACM/IEEE, 2017.

[59] G. Grassi, D. Pesavento, G. Pau, R. Vuyyuru, R. Wakikawa, L. Zhang. Vanet via named data networking. *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, 410–415. IEEE, 2014.

[60] G. Grassi, D. Pesavento, G. Pau, L. Zhang, S. Fdida. Navigo: Interest forwarding by geolocations in vehicular named data networking. *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, 1–10. IEEE, 2015.

[61] G. Grassi, D. Pesavento, L. Wang, G. Pau, R. Vuyyuru, R. Wakikawa, L. Zhang. Acm hotmobile 2013 poster: vehicular inter-networking via named data. *ACM SIGMOBILE Mobile Computing and Communications Review*, **17**(3), 23–24, 2013.

[62] D. Han, M. Lee, K. Cho, T. Kwon, Y. Choi. Publisher mobility support in content centric networks. *Proc. of ICOIN*, 2014.

[63] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, W. Seelen. An image processing system for driver assistance. *Image and Vision Computing*, **18**(5), 367–376, 2000.

[64] F. Hermans, E. Ngai, P. Gunningberg. Global source mobility in the content-centric networking architecture. *Proc. of ACM NoM Workshop*, 2012.

[65] A. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, L. Wang. Nlsr: named-data link state routing protocol. *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, 15–20. ACM, 2013.

[66] IEEE P802.11p TM/D3.0 Draft Standard for Information Technology - Amendment 7: Wireless Access in Vehicular Environments. *ANSI/IEEE Std 802.11, 1999 Edition (R2007)*. IEEE 802.11 working group, July 2007.

[67] IEEE802-WG. IEEE 802.11, part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *ANSI/IEEE Std 802.11, 1999 Edition (R2007)*, 2007.

[68] S. Ioffe, D. Forsyth. Human tracking with mixtures of trees. *Proc. IEEE Conf. ICCV*, vol. 1, 690–695, 2001.

[69] R. Itu, R. Danescu. An efficient obstacle awareness application for android mobile devices. *IEEE Conf. ICCP*, 157–163, 2014.

[70] ITU-R. Propagation data and prediction methods for the planning of short-range outdoor radiocommunication systems and radio local area networks in the frequency range 300 mhz to 100 ghz. Recommendation p.1441-9, International Telecommunication Union, Geneva, 2017.

[71] V. Jacobson, *et al.* Networking named content. *Proc. of CoNEXT*, 2009.

[72] M. Jerbi, S.-M. Senouci, T. Rasheed, Y. Ghamri-Doudane. Towards efficient geographic routing in urban vehicular networks. *IEEE Transactions on Vehicular Technology*, **58**(9), 5048–5059, 2009.

[73] R. Jiang, R. Klette, T. Vaudrey, S. Wang. New lane model and distance transform for lane detection and tracking. *CAIP*, 1044–1052. Springer, 2009.

[74] X. Jiang, J. Bi, Y. Wang. What benefits does NDN have in supporting mobility. *Proc. of IEEE ISCC*, 2014.

[75] D. B. Johnson, D. A. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile computing*, 153–181, 1996.

[76] B. Karp, H.-T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. *Proceedings of the 6th annual international conference on Mobile computing and networking*, 243–254. ACM, 2000.

[77] D.-h. Kim, J.-h. Kim, Y.-s. Kim, H.-s. Yoon, I. Yeom. Mobility support in content centric networks. *Proc. of ACM ICN 2012*.

[78] D.-h. Kim, J.-h. Kim, Y.-s. Kim, H.-s. Yoon, I. Yeom. End-to-end mobility support in content centric networks. *International Journal of Communication Systems*, **28**(6), 1151–1167, 2015.

[79] Y.-B. Ko, N. H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *Wireless networks*, **6**(4), 307–321, 2000.

[80] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, I. Stoica. A data-oriented (and beyond) network architecture. *ACM SIGCOMM Computer Communication Review*, vol. 37, 181–192. ACM, 2007.

[81] G. Korkmaz, E. Ekici, F. Özgüner, Ü. Özgüner. Urban multi-hop broadcast protocol for inter-vehicle communication systems. *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, 76–85. ACM, 2004.

[82] E. Koukoumidis, L.-S. Peh, M. R. Martonosi. Signalguru: leveraging mobile phones for collaborative traffic signal schedule advisory. *Proceedings of the 9th international conference on Mobile systems, applications, and services*, 127–140. ACM, 2011.

[83] D. Krajzewicz, J. Erdmann, M. Behrisch, L. Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, **5**(3&4), 128–138, 2012.

[84] G. Kreitz, F. Niemela. Spotify–large scale, low latency, p2p music-on-demand streaming. *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, 1–10. IEEE, 2010.

[85] X. Lan, D. P. Huttenlocher. Beyond trees: Common-factor models for 2d human pose recovery. *IEEE Conf. ICCV*, vol. 1, 470–477, 2005.

[86] N. D. Lawrence, A. J. Moore. Hierarchical gaussian process latent variable models. *Proc. of conf. on Machine learning*, 481–488. ACM, 2007.

[87] J. Lee, S. Cho, D. Kim. Device mobility management in content-centric networking. *Communications Magazine, IEEE*, **50**(12), 28–34, 2012. ISSN 0163-6804. doi:10.1109/MCOM.2012.6384448.

[88] K. C. Lee, J. Härri, U. Lee, M. Gerla. Enhanced perimeter routing for geographic forwarding protocols in urban vehicular scenarios. *Globecom Workshops, 2007 IEEE*, 1–10. IEEE, 2007.

[89] K. C. Lee, M. Le, J. Harri, M. Gerla. Louvre: Landmark overlays for urban vehicular routing environments. *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, 1–5. IEEE, 2008.

[90] K. C. Lee, U. Lee, M. Gerla. To-go: Topology-assist geo-opportunistic routing in urban vehicular grids. *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*, 11–18. IEEE, 2009.

[91] V. Lehman, A. Gawande, B. Zhang, L. Zhang, R. Aldecoa, D. Krioukov, L. Wang. An experimental investigation of hyperbolic routing with a smart forwarding plane in ndn. *Quality of Service (IWQoS), 2016 IEEE/ACM 24th International Symposium on*, 1–10. IEEE, 2016.

[92] B. Leibe, E. Seemann, B. Schiele. Pedestrian detection in crowded scenes. *IEEE Conf. CVPR*, vol. 1, 878–885, 2005.

[93] D. Li, M. C. Chuah. SCOM: A Scalable Content Centric Network Architecture with Mobility Support. *Proc. of IEEE MSN*, 2013.

[94] F. Li, Y. Wang. Routing in vehicular ad hoc networks: A survey. *IEEE Vehicular technology magazine*, **2**(2), 2007.

[95] R. Lienhart, J. Maydt. An extended set of haar-like features for rapid object detection. *Proc. Conf. on Image Processing*, vol. 1, I–900. IEEE, 2002.

[96] T. D. Little, A. Agarwal, *et al.* An information propagation scheme for vanets. *Proc. IEEE Intelligent Transportation Systems*, 155–160, 2005.

[97] G. Liu, B.-S. Lee, B.-C. Seet, C.-H. Foh, K.-J. Wong, K.-K. Lee. A routing strategy for metropolis vehicular communications. *Information networking. networking technologies for broadband and mobile networks*, 134–143, 2004.

[98] X. Liu, M. J. Nicolau, A. Costa, J. Macedo, A. Santos. A geographic opportunistic forwarding strategy for vehicular named data networking. *Intelligent Distributed Computing IX*, 509–521. Springer, 2016.

[99] C. Lochert, H. Hartenstein, J. Tian, H. Fussler, D. Hermann, M. Mauve. A routing strategy for vehicular ad hoc networks in city environments. *Intelligent Vehicles Symposium, 2003. Proceedings. IEEE*, 156–161. IEEE, 2003.

[100] C. Lochert, M. Mauve, H. Füßler, H. Hartenstein. Geographic routing in city scenarios. *ACM SIGMOBILE mobile computing and communications review*, **9**(1), 69–72, 2005.

[101] S. Ma, O. Wolfson, B. Xu. Updetector: Sensing parking/unparking activities using smartphones. *Proceedings of the 7th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, 76–85. ACM, 2014.

[102] R. Mahmud, R. Buyya. Fog computing: A taxonomy, survey and future directions. *arXiv preprint arXiv:1611.05539*, 2016.

[103] C. Maihofer, R. Eberhardt. Geocast in vehicular environments: caching and transmission range control for improved efficiency. *Intelligent Vehicles Symposium, 2004 IEEE*, 951–956. IEEE, 2004.

[104] H. A. Mallot, H. H. Bülthoff, J. Little, S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological cybernetics*, **64**(3), 177–185, 1991.

[105] J. G. Manweiler, P. Jain, R. Roy Choudhury. Satellites in our pockets: an object positioning system using smartphones. *MobiSys*, 211–224. ACM, 2012.

[106] S. Mastorakis, A. Afanasyev, I. Moiseenko, L. Zhang. ndnSIM 2: An updated NDN simulator for NS-3. Technical Report NDN-0028, Revision 2, NDN, 2016.

[107] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, W. Trappe. Parknet: drive-by sensing of road-side parking statistics. *MobiSys*, 123–136. ACM, 2010.

[108] A. Mcgregor, D. Smithies. Rate adaptation for 802.11 wireless networks: Minstrel. *Submitted to ACM SIGCOMM 2010, http://blog.cerowrt.org/papers/minstrel-sigcomm-final.pdf.*

[109] ETSI, Mobile-Edge Computing Introductory Technical White Paper. `http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing`.

[110] IBM News Releases, "IBM and Nokia Siemens Networks announce world first mobile edge computing platform". `http://www-03.ibm.com/press/us/en/pressrelease/40490.wss`.

[111] V. Namboodiri, M. Agarwal, L. Gao. A study on the feasibility of mobile gateways for vehicular ad-hoc networks. *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, 66–75. ACM, 2004.

[112] A. Nandugudi, T. Ki, C. Nuessle, G. Challen. Pocketparker: Pocketsourcing parking lot availability. *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 963–973. ACM, 2014.

[113] S. Nawaz, C. Efstratiou, C. Mascolo. Parksense: A smartphone based sensing system for on-street parking. *Proceedings of the 19th annual international conference on Mobile computing & networking*, 75–86. ACM, 2013.

[114] V. D. Nguyen, T. T. Nguyen, D. D. Nguyen, J. W. Jeon. Toward real-time vehicle detection using stereo vision and an evolutionary algorithm. *VTC Spring*, 1–5. IEEE, 2012.

[115] Y. Nishiyama, M. Ishino, Y. Koizumi, T. Hasegawa, K. Sugiyama, A. Tagami. Proposal on routing-based mobility architecture for icn-based cellular networks. *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*, 467–472. IEEE, 2016.

[116] D. Noll, M. Werner, W. Von Seelen. Real-time vehicle tracking and classification. *Proc. of the Intelligent Vehicles Symposium*, 101–106. IEEE, 1995.

[117] C.-C. Ooi, N. Fisal. Implementation of geocast-enhanced aodv-bis routing protocol in manet. *TENCON 2004. 2004 IEEE Region 10 Conference*, 660–663. IEEE, 2004.

[118] OpenCV: `http~://opencv.org/`.

[119] OpenFog Architecture Overview. Open Fog Consortium. February 2016. `https://www.openfogconsortium.org/wp-content/uploads/OpenFog-Architecture-Overview-WP-2-2016.pdf`.

[120] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, A. V. Vasilakos. Fog computing for sustainable smart cities: A survey. *arXiv preprint arXiv:1703.07079*, 2017.

[121] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos. Sensing as a service model for smart cities supported by internet of things. *Transactions on Emerging Telecommunications Technologies*, **25**(1), 81–93, 2014.

[122] Periscope. Video Streaming, https://www.periscope.tv/.

[123] C. Perkins, E. Belding-Royer, S. Das. Ad hoc on-demand distance vector (aodv) routing. Tech. rep., 2003.

[124] R. Ravindran, S. Lo, X. Zhang, G. Wang. Supporting seamless mobility in named data networking. *Proc. of IEEE ICC*, 2012.

[125] F. Ren, J. Huang, M. Terauchi, R. Jiang, R. Klette. Lane detection on the iphone. *International Conference on Arts and Technology*, 198–205. Springer, 2009.

[126] M. Rezaei, M. Terauchi, R. Klette. Robust vehicle detection and distance estimation under challenging lighting conditions. *Transactions on Intelligent Transportation Systems*, 2015.

[127] J. W. Roberts. Realizing quality of service guarantees in multiservice networks. *Performance and Management of Complex Communication Networks*, 277–293. Springer, 1998.

[128] R. Roman, J. Lopez, M. Mambo. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, 2016.

[129] A. Rowstron, G. Pau. Characteristics of a vehicular network. *University of California Los Angeles, Computer Science Department, Tech. Rep*, 09–0017, 2009.

[130] R. A. Santos, A. Edwards, R. Edwards, N. L. Seed. Performance evaluation of routing protocols in vehicular ad-hoc networks. *International Journal of Ad Hoc and Ubiquitous Computing*, **1**(1-2), 80–91, 2005.

[131] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, **8**(4), 2009.

[132] C. Scientific, I. R. O. (CSIRO). Phenonet: Dis- tributed sensor network for phenomics supported by high resolution plant phenomics centre. *CSIRO ICT Centre, and CSIRO Sensor and Sensor Networks TCP*, 2011.

[133] E. Seemann, B. Schiele. Cross-articulation learning for robust detection of pedestrians. *Pattern Recognition*, 242–252. Springer, 2006.

[134] Y. Sekimoto, Y. Matsubayashi, H. Yamada, R. Imai, T. Usui, H. Kanasugi. Lightweight lane positioning of vehicles using a smartphone gps by monitoring the distance from the center line. *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, 1561–1565. IEEE, 2012.

[135] I. Seskar, K. Nagaraja, S. Nelson, D. Raychaudhuri. Mobilityfirst future internet architecture project. 1–3, 2011.

[136] SFPark: `http~://sfpark.org/`.

[137] S. Shenker. Fundamental design issues for the future internet. *Selected Areas in Communications, IEEE Journal on*, **13**(7), 1176–1188, 1995.

[138] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, **3**(5), 637–646, 2016.

[139] D. C. Shoup. Cruising for parking. *Transport Policy*, **13**(6), 479–486, 2006.

[140] L. Sigal, M. J. Black. Measure locally, reason globally: Occlusion-sensitive articulated pose estimation. *IEEE Conf. CVPR*, vol. 2, 2041–2048, 2006.

[141] S. Soubam, D. Banerjee, V. Naik, D. Chakraborty. Bluepark: tracking parking and un-parking events in indoor garages. *Proceedings of the 17th International Conference on Distributed Computing and Networking*, 33. ACM, 2016.

[142] Streetline Networks: `http~://www.streetlinenetworks.com/`.

[143] M.-T. Sun, W.-C. Feng, T.-H. Lai, K. Yamada, H. Okada, K. Fujimura. Gps-based message broadcasting for inter-vehicle communication. *Parallel Processing, 2000. Proceedings. 2000 International Conference on*, 279–286. IEEE, 2000.

[144] Z. Sun, G. Bebis, R. Miller. On-road vehicle detection: A review. *IEEE TPAMI*, **28**(5), 694–711, 2006.

[145] P. TalebiFard, V. Leung. A content centric approach to dissemination of information in vehicular networks. *Proceedings of the second ACM international symposium on Design and analysis of intelligent vehicular networks and applications*, 17–24. ACM, 2012.

[146] Yellow cab of san francisco, location dataset. `http://cabspotting.org/`.

[147] D. Tian, K. Shafiee, V. C. Leung. Position-based directional vehicular routing. *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, 1–6. IEEE, 2009.

[148] Time Warner Cable. TWC WiFi coverage map, http://coverage.twcwifi.com/.

[149] G. Toulminet, M. Bertozzi, S. Mousset, A. Bensrhair, A. Broggi. Vehicle detection by means of stereo vision-based obstacles features extraction and monocular pattern analysis. *IEEE Transactions on Image Processing*, **15**(8), 2364–2375, 2006.

[150] N. True. Vacant parking space detection in static images. *University of California, San Diego*, 2007.

[151] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, **3**(4), 323–344, 1987.

[152] S. Tuohy, D. O'Cualain, E. Jones, M. Glavin. Distance determination for an automobile environment using inverse perspective mapping in opencv. *Proc. ISSC*, 2010.

[153] G. Tyson, A. Mauthe, S. Kaune, P. Grace, T. Plagemann. Juno: An adaptive delivery-centric middleware. 587–591, 2012.

[154] G. Tyson, N. Sastry, R. Cuevas, I. Rimac, A. Mauthe. A survey of mobility in information-centric networks. *Commun. ACM*, **56**(12), 90–98, 2013.

[155] G. Tyson, N. Sastry, I. Rimac, R. Cuevas, A. Mauthe. A survey of mobility in information-centric networks: challenges and research directions. *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications*, 1–6. ACM, 2012.

[156] R. Urtasun, D. J. Fleet, P. Fua. 3d people tracking with gaussian process dynamical models. *IEEE Conf. CVPR*, vol. 1, 238–245, 2006.

[157] L. M. Vaquero, L. Rodero-Merino. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review*, **44**(5), 27–32, 2014.

[158] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, D. S. Nikolopoulos. Challenges and opportunities in edge computing. *Smart Cloud (SmartCloud), IEEE International Conference on*, 20–26. IEEE, 2016.

[159] R. Viereckl, D. Ahlemann, A. Koster, et al. Connected car report 2016: Opportunities, risk, and turmoil on the road to autonomous vehicles., 2016.

[160] P. Viola, M. Jones. Rapid object detection using a boosted cascade of simple features. *Proc. of IEEE Conf. CVPR*, vol. 1, I–511. IEEE, 2001.

[161] C. Wah. Parking space vacancy monitoring. *Projects in Vision and Learning*, 2009.

[162] J. Wang, R. Wakikawa, L. Zhang. Dmnd: Collecting data from mobiles using named data. *Vehicular Networking Conference (VNC), 2010 IEEE*, 49–56. IEEE, 2010.

[163] L. Wang, A. Afanasyev, R. Kuntz, R. Vuyyuru, R. Wakikawa, L. Zhang. Rapid traffic information dissemination using named data. *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications*, 7–12. ACM, 2012.

[164] L. Wang, A. Hoque, C. Yi, A. Alyyan, B. Zhang. Ospfn: An ospf based routing protocol for named data networking, 2012.

[165] L. Wang, R. Wakikawa, R. Kuntz, R. Vuyyuru, L. Zhang. Data naming in vehicle-to-vehicle communications. *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, 328–333. IEEE, 2012.

[166] L. Wang, O. Waltari, J. Kangasharju. Mobiccn: Mobility support with greedy routing in content-centric networks. *Proc. of IEEE GLOBECOM*, 2013.

[167] A. Wedel, U. Franke, J. Klappstein, T. Brox, D. Cremers. Realtime depth estimation and obstacle detection from monocular video. *Pattern Recognition*, 475–484. Springer, 2006.

[168] T. A. Williamson. *A high-performance stereo vision system for obstacle detection.* Ph.D. thesis, Carnegie Mellon, 1998.

[169] B. Wu, R. Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, **75**(2), 247–266, 2007.

[170] United Nations. World Urbanization Prospects, 2014 Revision. `https://esa.un.org/unpd/wup/Publications/Files/WUP2014-Highlights.pdf`.

[171] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, G. C. Polyzos. A survey of information-centric networking research. *IEEE Communications Surveys & Tutorials*, **16**(2), 1024–1049, 2014.

[172] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, L. Zhang. A Case for Stateful Forwarding Plane. *Computer Communications: ICN Special Issue*, **36**(7), 779–791, 2013.

[173] R. Ying, L. Hongbin, G. Deyun, Z. Huachun, Z. Hongke. Lbma: A novel locator based mobility support approach in named data networking. *China Communications*, **11**(4), 111–120, 2014.

[174] C.-W. You, N. D. Lane, *et al.* Carsafe app: alerting drowsy and distracted drivers using dual cameras on smartphones. *MobySys*, 13–26. ACM, 2013.

[175] Y. Yu, A. Afanasyev, D. Clark, V. Jacobson, L. Zhang, *et al.* Schematizing trust in named data networking. *Proceedings of the 2nd International Conference on Information-Centric Networking*, 177–186. ACM, 2015.

[176] Y.-T. Yu, Y. Li, X. Ma, W. Shang, M. Sanadidi, M. Gerla. Scalable opportunistic vanet content routing with encounter information. *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, 1–6. IEEE, 2013.

[177] Y.-T. Yu, T. Punihaole, M. Gerla, M. Sanadidi. Content routing in the vehicle cloud. *Military Communications Conference, 2012-milcom 2012*, 1–6. IEEE, 2012.

[178] Y. Yu-Ting, C. Tandiono, X. Li, Y. Lu, M. Sanadidi, M. Gerla. Ican: Information-centric context-aware ad-hoc network. *Computing, Networking and Communications (ICNC), 2014 International Conference on*, 578–582. IEEE, 2014.

[179] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Méndez, C. E. Chung, Y. Te Wang, T. Mullen, T. P. Jung. Augmented brain computer interaction based on fog computing and linked data. *Intelligent Environments (IE), 2014 International Conference on*, 374–377. IEEE, 2014.

[180] Y. Zhang, A. Afanasyev, J. Burke, L. Zhang. A survey of mobility support in named data networking. *Proc. of IEEE INFOCOM NOM*, 2016.

[181] Y. Zhang, H. Zhang, L. Zhang. Kite: A mobility support scheme for ndn. *Proc. of ACM ICN Poster*, 2014.

[182] Z. Zhang. A flexible new technique for camera calibration. *IEEE TPAMI*, **22**(11), 1330–1334, 2000.

[183] Z. Zhu, R. Wakikawa, L. Zhang. A survey of mobility support in the internet. *RFC 6301*, 2011.