University of California

Los Angeles

Information-Centric Vehicular Ad-Hoc Networks: Challenges and Solutions

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Yu-Ting Yu

2014

Abstract of the Dissertation

Information-Centric Vehicular Ad-Hoc Networks: Challenges and Solutions

by

Yu-Ting Yu

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2014

Professor Mario Gerla, Co-chair

Professor Mohammad Yahya Sanadidi, Co-chair

Recently, Information-Centric Networking (ICN) has attracted much attention in part because of its promising future as next-generation Internet architecture. While ICN is scalable and efficient in the Internet, it raises concerns when deployed in a mobile large scale network like the Vehicular Ad-hoc Network (VANET). For example, conventional ICN techniques do not work well in the intermittent VANET connectivity. Moreover, current ICN proposal strictly follows a receiver-driven transport design. However, many applications in multi-hop adhoc networks are push-based and require fast communications. ICN's pull-based transport in such cases is underperforming and costly. Finally, the inefficiency introduced by ICN default anycast may backfire without careful design.

In this study, we introduce Information-Centric Ad-hoc Network (ICAN), an efficient, flexible, and adaptive ICN architecture supporting both pull and push transport and contextaware, multi-hop and disruption-tolerant communications all in one system.

The dissertation of Yu-Ting Yu is approved.

University of California, Los Angeles

2014

Table of Contents

List of Figures

List of Tables

Acknowledgments

My immense gratitude goes to my advisor, Dr. Mario Gerla, who guided and inspired me in every aspects during my doctoral study. He is the best and most encouraging advisor I have ever known. The depth and breadth of this study would not have been achieved without his everlasting encouragement and advice.

I would like to thank Dr. M. Y. Sanadidi, who offered insightful advice and enduring patience at the beginning of my study. He has been always helpful and encouraging. I also appreciate the other members in my doctoral committee, Dr. Milos Ercegovac, Dr. Danijela Cabric, and Dr. Jack Carlyle, for their insight and advice making this work deeper and more complete than it would otherwise have been.

A special appreciation goes to every NRL member. I thank Dr. Chien-Chia Chen, Dr. Giovanni Pau, Mr. Joshua Joy, Mr. Ruolin Fan, Mr. Xiao Li, and Mr. You Lu for their assistance on our collaborative works. Special appreciation goes to Mr. Aik Tuan Lee, Mr. Fabio Angius, and Dr. Jui-Ting Weng for their friendship and support. I also thank all my project partners, Mr. Chris Tandiono, Mr. Denis Lu, Miss Hsuan Chiu, Mr. Victor Perez, Mr. Wentao Shang, Mr. Xingyu Ma, and Mr. Yuanjie Li. Without their help, many ideas would have taken much longer to be implemented.

This study would not been possible without the support of my family. No words can express my gratitude to my parents, and all my family members in Taiwan. My deepest gratitude goes to Dr. Chi Cheng, for her long lasting support. I appreciate all my friends, who accompanied me during this long journey at UCLA. This list cannot be comprehensive. My gratitude goes to all of you.

Finally, this study could not have been done without the support of our sponsors, NSF, Singapore DSO National Laboratories, and Stanford Research Institute.

Vita

2006	B.S. (Computer Science), National Chiao Tung University.
2012	M.S. (Computer Science), UCLA, Los Angeles, California.

Publications

Y.-T. Yu, M. Gerla, M. Y. Sanadidi, "Scalable VANET Content Routing Using Hierarchical Bloom Filters", accepted by Wireless Communications And Mobile Computing, 2014, DOI: 10.1002/wcm.2495

J. Joy, Y.-T. Yu, V. Perez, D. Lu, M. Gerla, "A New Approach to Coding in Content-Based MANETs", accepted by Journal of Communication, 2014

Y.-T. Yu, C. Tandiono, X. Li, Y. Lu, M. Y. Sanadidi, M. Gerla, "ICAN: Information-Centric Context-Aware Ad-Hoc Network", in Proceedings of IEEE ICNC'14, February 2014, Honolulu, Hawaii.

J. Joy, Y.-T. Yu, V. Perez, D. Lu, M. Gerla, "A New Approach to Coding in Content-Based MANETs", in Proceedings of IEEE ICNC'14, February 2014, Honolulu, Hawaii.

Y.-T. Yu, Y. Li, W. Shang, X. Ma, M. Y. Sanadidi, and M. Gerla, "Scalable Opportunistic VANET Content Routing with Encounter Information," in Proceedings of VCA'13, in conjuction with IEEE ICNP13, Oct 2013, Gottingen, Germany.

Y.-T. Yu, X. Li, M. Gerla, and M.Y. Sanadidi, "Scalable VANET Content Routing Using Hierarchical Bloom Filters, " in Proceedings of IEEE IWCMC'13, July 2013, Cagliari, Italy.

Y.-T. Yu, R. Dilmaghani, S. Calo, M. Y. Sanadidi, and M. Gerla "Interest Propagation In Named Data MANETs," in Proceedings of IEEE ICNC'13, January 2013, San Diego, CA, USA.

Y.-T. Yu, T. Punihaole, M. Gerla, M. Y. Sanadidi, "Content Routing In The Vehicle Cloud," in Proceedings of IEEE MILCOM'12, Octobor 2012, Orlando, FL, USA.

C. -C. Chen, G. Tahasildar, Y.-T. Yu, J.-S. Park, M. Gerla, M. Y. Sanadidi, "CodeMP: Network Coded Multipath to Support TCP in Disruptive MANETs," in Proceedings of IEEE MASS'12, Octobor 2012, Las Vegas, NV, USA

J. Joy, Y.-T. Yu, Mario Gerla, S. Wood, J. Mathewson, and M.-O. Stehr, "Reliable Dissemination of Large Objects using Network Coding", Mobicom'13, October 2013, Miami, FL.

Y.-T. Yu, J. Joy, M. Gerla, and M. Y. Sanadidi, "DT-ICAN: A Delay-tolerant Information-Centric Ad-Hoc Network," #TD130017, Computer Science Department, University of California, Los Angeles, 2013.

CHAPTER 1

Introduction

Today's mobile networking very much relies on the infrastructure technology. However, the advance of infrastructure deployment cannot always meet the bursting demands of diverse scenarios and applications on the user end. To meet the growing demands of vehicular networking, mobile ad-hoc networking with the advantages of fast deployment and easy device replacement is attracting more and more attentions. Ad-hoc communication enables the possibility of communication in rural or emergency scenarios that lack of infrastructure coverage, and also enables various possibilities for applications that were not usable or reliable in the infrastructure mobile network. For example, a number of applications utilizing direct machine to machine connectivity such as autonomous driving vehicles [2][3], urban surveillance [4][5], and content sharing [6] have been broadly discussed.

As the demands of highly mobile ad-hoc networking increase in the future, it is critical to propose an architectural mechanism rather than point solutions for the ease of rapid on-demand ad-hoc network deployment. To this end, we introduce Information-Centric Context-aware Ad-hoc Networking (ICAN), a scalable machine-to-machine networking architecture that is flexible and adaptive to diverse application requirements and network conditions. While it is difficult to define a one-fits-all architecture, in ICAN, the application needs and network conditions are compiled as context in an extendable format so that it is possible to customize the ad-hoc networking platform in a software controllable manner for quick deployment and management.

The ICAN architecture builds on top of three building blocks: network entity representation, context, and network operation. The network entity representation defines the identifier of communication units flowing in the network such as hosts, data objects, and data chunks. Network entities can be associated with each other. One example is to use the hierarchical namespace broadly used by today's ICN community, where a file is identified by its file name with a prefix identifying its host ID, and is segmented as chunks each with a chunk ID [7]. Once the network entities are identified, their associated contexts are built automatically. We define the context be either application-related (e.g. content type) or network condition-related (e.g. connectivity). The context is leveraged by the network operations to adaptively decide the packet forwarding and caching strategies.

Like most architectural proposals, the components of ICAN are inspired by prior work, including Information Centric Networking (ICN), Disruption Tolerant Networking (DTN), and opportunistic routing [8]. Our core contribution is extending and integrating them in a new architecture. For example, we adopt the concept of ICN [7], which enforces the receiver-oriented chunk based transport and in-network caching. While in-network caching is beneficial under mobility, many ad-hoc network applications are substantially push-based. Although such applications are also achievable with pull-based transport, the fundamental inefficiency of pull-only designs will degrade or even destroy the performance in VANET due to the limited bandwidth and storage. Therefore, we include in ICAN the push-based transport as a fundamental unit and leverage both push and pull paradigms by application context.

Although ICAN itself is a common framework suitable for all ad-hoc network and also applicable to infrastructure network, in this study, we focus particularly in scenarios such as urban vehicular network and emergency network constituted by human rescuers and highly mobile vehicles, as we envision that the pervasive caching and context awareness of ICN have highest potential in highly mobile networks.

The rest of this study is organized as follows. In chapter 2, we briefly discuss the background and related literature. In chapter 3, we introduce the ICAN system architecture. In chapter 4, we study the design options on routing and caching for urban VANET using extensive simulation. With the insights gained from the simulation, in chapter 5, we propose a context-aware content discovery for urban vehicular ICN. In chapter 6, we examine the ICAN system in sparse emergency VANET, and introduce the network coding extension to improve the efficiency. Finally, we conclude in chapter 7.

CHAPTER 2

Background

2.1 Information Centric Networks

2.1.1 ICN Background

Our proposed architecture, ICAN, relies heavily on the concept of ICN, also known as Named Data Networks (NDN) or Content-Centric Networks (CCN) [7], which is a prominent ICN design aiming for replacing TCP/IP. ICN uses the data names instead of host addresses to locate data. Every data chunk has a unique name. To initiate a data transfer, a data consumer must send an Interest to retrieve the corresponding Data. In other words, ICN strictly assumes a pull-based, one-interest-one-data transport. The data chunks are cached along the way by all relays when it traverses the breadcrumb of interest forwarding path from a data provider back to the data consumer. The ICN packet format is shown in Figure 2.1.

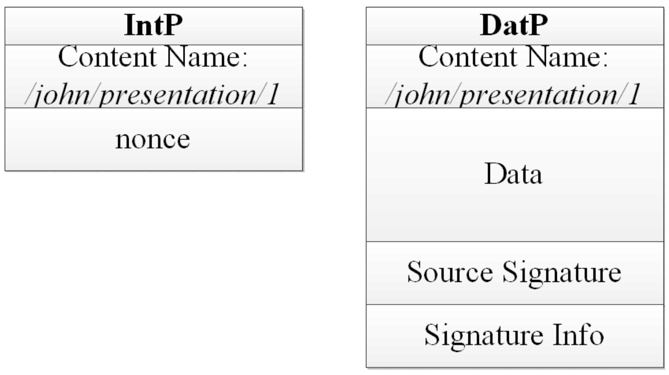

Figure 2.1: ICN packet format

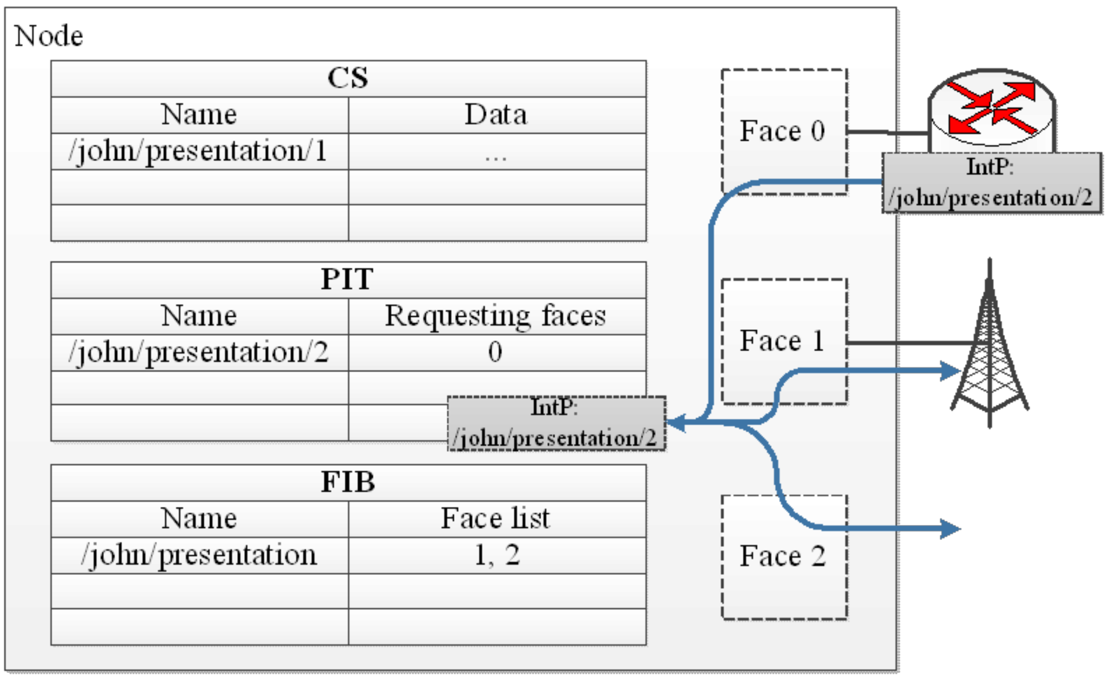

Figure 2.2: ICN node model

All ICN nodes are identically built with three data structures: Content Store (CS), Pending Interest Table (PIT), and Forwarding Information Base (FIB). An ICN node model is shown in Figure 2.2. CS is a cache used to store received data. The use of CS is the core design of ICN and the main reason why we choose to build ICAN as an extended ICN architecture. Caching eases issues caused by unreliable wireless channel. With distributed caches, a data retrieval failure arising from intermittent connectivity can be quickly recovered.

PIT "remembers" all pending interests a node received. The two main purposes of PIT are (1) recording the breadcrumb path, and (2) suppressing redundant interest and data transmission for different data consumers. The above usage of PIT is sufficient to achieve efficient Internet routing since nearly all multi-hop routing protocols use a breadcrumb path of the exploration packet. However, PIT alone is not enough for reliable and efficient data delivery in ad-hoc networks. Under intermittent connectivity, the name-based routing must be adjusted so that data can be delivered in a carry-and-forward way. On the other hand, interest aggregation is valuable as bandwidth utilization can be largely improved. Therefore, there is a need to extend and re-define PIT's functionality. FIB is a routing table constructed per name prefix. ICN interests are broadcast on one or more interface recorded in the routing table. Using broadcast is advantageous due to the fact that more caches can be explored. However, the table-driven routing does not always fit needs in ad-hoc networks and thus also requires a re-definition.

The above design follows a late-binding approach in which no name-locator mapping is handled before the interest is sent. Data consumer mobility is well-handled when the end-to-end connectivity is relatively stable in such design due to its one-interest-one-data nature. A data consumer can re-initiate interests to obtain the data cached by relays when it relocates. However, data producer mobility is more difficult to conquer as it is assumed that prefix naming is based on organization names which are bound to relatively static locations. ICN may mitigate effects of data producer mobility by multi-sourcing (i.e. relying on multiple previously established caches), but this naive design may be problematic in vehicular networks. In typical VANET deployment where a node uses only one interface, the interface-selecting broadcast design results in all nodes flooding all Interest Packets they received. When data producers and relays are moving, relying on interest flooding may lead to data storm, which is induced by duplicate cached data from multiple data holders. Therefore, the flooding traffic will lead to extreme bandwidth consumption. One solution is to construct an overlay network on top of IP layer. However, implementing point-to-point unicast overlay ICN is costly in VANET because (1) end-to-end route construction and maintenance between overlay nodes induce high control overhead and (2) the overlay design ends up performing point-to-point transmission without exploiting the broadcasting nature of wireless channel. In fact, it is well known that the routing table-driven protocols are not suitable under high mobility [9]. In ICAN, we exploit opportunistic routing [8] to utilize wireless broadcast nature for better robustness.

2.1.2 Alternative ICN designs

Alternative ICN designs have also been proposed and discussed. For example, Data-Oriented Network Architecture (DONA)[10] is another well-known content-oriented framework which is defined as an overlay over IP network. In DONA, a content name is formed as cryptographic hash of the data producer's public key following by a data object identifier. Unlike NDN, DONA takes an early-binding approach in which data producers register the name-locator mapping to servers called Resolution Handlers (RHs). The RHs form a tree structure similar to the DNS system, and data is searched by lookup following the tree structure. Data producer mobility is easier to resolve in DONA since the name-locator mapping must be re-published when data producers move. However, the challenges in VANET are the significant re-publishing overhead and the time and space complexities to maintain RH tree structures. Data consumer mobility can create problems in DONA as well. Mobile data consumers must re-establish connections to new RHs as they move, and consequently introduce larger delay and overhead for data retrieval. Being implemented on top of TCP/IP, current DONA implementation also requires significant modifications to be applied to VANET.

Network of Information (NetInf) [11] also utilizes a name-resolution approach. NetInf assumes flat naming. Data producers publish the contents and their locators via the Name Resolution (NR) service. Different from DONA, NetInf constructs the NR service by a Multi-level DHT (MDHT) [12] supporting both data and metadata searching. NetInf also requires NR service connection re-establishment under data consumer mobility and locator updates under data source mobility.

Although the existing ICN proposals differ in details such as the use of opaque content identifier [10] or human readable hierarchical name [7], we identify the following common attributes of ICN:

- Uniquely identifiable content naming: Different from the host-based model, ICN decouples the contents with locations to better suit application needs and to enable caching. All ICN proposals assume contents are uniquely identifiable at network level so that the network can identify the content to be transmitted.

- In-network caching: The named contents enable the pervasive in-network caching. In the extreme cases, all routers may cache content segments flow through. The caching capability of ICN is naturally creates a multi-source environment and thus is potentially beneficial under mobility.

- Receiver-driven transport: As contents are available at multiple locations, the idea of a particular network host serving certain contents becomes obscure. In such setting, the content consumer, or the receiver, must actively retrieve and control the content transfer.

- Name-based forwarding: In ICN, content is searched and retrieved based on its identity instead of the IP address of the node on which it resides. The interest forwarding of a content request is performed based on the content names. This gives the network the ambiguity of selecting the destination a content request to be sent to. In most ICN proposals, it is assumed the request is forwarded to the nearest replica.

- Built-in security: the names are bound to the intent of the publisher, and thus help ease the design of security mechanism such as integrity checking on the user side.

In this paper, we focus on analyzing the general benefits of ICN in ad-hoc networks. For simplicity, we choose to build ICAN as an extension of [7] as (1) it is the most mature ICN architecture to date, and (2) it depends the least on centralized controller in the original design and thus is the easiest to adapt to ad-hoc networks.

2.1.3 Mobile ICN

ICN in mobile environments is a recently emerging research area. For mobile networks with infrastructure support, J. Wang et al. propose an NDN-based vehicle data collection system in [13]. This system requires special name prefixes to be reserved and announced in advance. J. Lee et al. propose a proxy-based scheme to increase the efficiency of mobile retrievals [14].

As for MANETs, in [15], S. Oh et al. propose an overlay tactical ICN approach and study its feasibility and performance via implementation. In [16], M. Varvello et al. analyze the performance of reactive flooding, proactive flooding, and Geographic Hash Table (GHT) in ICN. Proactive flooding floods the data object names and locations periodically. GHT maintains the data object information at fixed locations that can be calculated by pre-defined hash functions. In contrast, reactive flooding naively floods all Interest Packets to retrieve Data Packets. The results in [16] analytically show that the reactive flooding approach outperforms the other two in both latency and data availability.

The ICN architecture proposed in [7] in its original form assumes the medium is reliable enough and the "faces" of transmissions are relatively stable, which are different from the nature of ad-hoc networks. Most existing ad-hoc ICN research propose to build a broadcast-only system to better utilize the wireless broadcast channel and minimize the control overhead under mobility. The first such proposal is [17]. In [17], Meisel et al. propose a low-overhead forwarding protocol for MANET ICN. Each node maintains distances to known data object names and node names. The requests are broadcast but propagated to the nearest data holder via shortest path using opportunistic routing [8] based on the destination distances estimated by relays. In [18], Amadeo et al. describe a multi-hop MANET ICN architecture in which the content consumer locates content provider using controlled flooding. Nodes keep track of the content provider IDs and use this information to perform consumer-based provider selection. The transmissions are broadcast but counter-based suppression is applied to reduce unnecessary transmissions. The extended works [19][20] examine similar system in VANET and show improvements on content delivery over IP-based VANET using AODV in a small scenario in which 5-25 consumers download contents from a road-side unit. However, the counter-based broadcast suppression approach may incur unnecessary transmissions and packet loss due to the interference and inaccurate node distance information under high mobility and heavier application traffic.

The state-of-art VANET ICN research mainly focus on reducing the control overhead of locating mobile content provider by methods that commonly applied to host-based VANET. In this study, we take one step forward and explore the common effects behind the potential system design options for multi-hop, connected ad-hoc networks. We start by assessing the caching benefits in vehicle-to-vehicle communications using synthetic user behavior model, and then continue the study of content discovery mechanisms according to our findings.

2.2 Ad-hoc routing

As an architectural solution, a critical aspect of ICAN design is the efficient routing mechanisms. In this study, we identify two representative VANET scenarios: urban VANET and emergency VANET. In urban VANET, the vehicle density is high while obstacles often block the transmissions, leading to a network with rich though sometimes intermittent end-to-end connectivities. In emergency VANET, the network is sparse and the connectivity is rare. End-to-end sessions are nearly impossible even in short term. In the following, we review the literature on the ad-hoc routing mechanisms for these scenarios.

2.2.1 MANET routing

The traditional IP-based multi-hop mobile ad-hoc routing can be divided to three categories: proactive routing, reactive routing, and opportunistic routing.

IP reactive routing consists of two phases: route discovery and route maintenance. Route discovery serves the purpose of locating destination host and is accomplished via flooding. A source node first floods a routing request to search the destination node. Once the destination is found, a routing reply is forwarded back to the source node via a single path with the lowest cost. The cost at each hop is collected in the process of request flooding. Data transmissions then take place using the single path constructed. Route maintenance is performed explicitly with routing control messages when a topology change is detected and the previously-built path is consequently broken. Popular reactive IP routing mechanism includes AODV [21], DSR[22], and their variations.

Proactive IP routing protocols such as Fisheye State Routing (FSR) [23] and Optimized Link State Routing Protocol (OLSR) [24] are based on proactive host advertising, utilizing periodic background routing information exchange. Based on the received routing information from other nodes, each node construct routing tables for each possible destination host with link-state routing [25] technique. The common advantage of this type of routing protocols is the low-latency route access and consquently QoS guarantee once the routing states converge. However, they also require excessive overhead for routing information exchange. Researchers have devoted hard efforts on reducing the advertisement overhead. For example, OLSR uses multipoint relays (MPR) [26] to reduce the number of superfluous broadcast transmissions.

Opportunistic routing such as Extremely Opportunistic Routing (ExOR) [8] defers the choice of the next hop until the forwarding set receiving the packet. It is a good example of exploiting wireless broadcast nature. Each node maintains its distance to destination based on a distance metric. Instead of choosing a static route, all packets are broadcast. Upon receiving a broadcast packet, nodes set up a backoff time proportional to its distance to the destination. During the backoff, if a node hears the packet being re-broadcasting by other nodes, it cancels the scheduled transmission of the same packet. Otherwise, the node re-broadcast the packet once the backoff ends. The advantage of such approach is its minimum control overhead since forwarder selection is deferred to the time of receipt. The major challenge comes from the accuracy of estimating the distance metric of each node.

2.2.2 VANET routing

VANET routing research shares the reactive and proactive routing ancestors from MANET research. In addition, geo-routing is the most promising routing method in VANET. This routing family utilizes the Global Positioning System (GPS) equipped by the host nodes to provide their precise location information, which is then utilized to calculate the distance to destination host. Based on the distance information, a forwarder set, which contains all nodes nearer the destination than the previous hop, can be defined, and packets approach the destination relying on the set of eligible forwarders hop-by-hop. The details of actual forwarder selecting vary for different protocols. In Location-Aided Routing (LAR) [27], the forwarder set is only used for the purpose of routing interest limited flooding. The data forwarding is based on the resulting shortest path constructed in a way similar to that of DSR. In Greedy Perimeter Stateless Routing (GPSR) [28], nodes periodically broadcast its GPS coordinate to one-hop neighbors. Packet senders are responsible for specifying the next hop IP address of the best forwarder who is closer to the destination. The common advantage of geo-routing protocols is the relatively small control overhead and consequently improved routing performance. However, additional care must be taken considering the potential location inaccuracy in a mobile environment.

2.2.3 Disruption tolerant routing

Disruption-Tolerant Networking (DTN) usually refers to networking when the scenario is sparse and thus it is difficult to obtain full topology information at each node. Most routing protocols in such environments are variations of Epidemic Routing (ER) [29]. ER diffuses messages into networks in a similar way as diseases using the cache-and-forward technique. To reduce the overhead and delay of ER, MV routing [30] opportunistically selects messages to forward to encountered nodes. Follow-up works [31][32] utilize mobility patterns or contact history to further improve the performance.

2.3 Network coding

It has been show that network coding is beneficial in cache-capable MANETs [33, 34]. Lee et al. showed that by exploiting dissemination of coded fragments of files, network coding is able to greatly decrease the delay required to deliver files. In [35], Montpetit et al. have identified network coding ICN as a strategy with tremendous potential. In [36] Wu et al. implemented and evaluated the benefits network coding provides in improving the cache hits in ICN.

MORE [37] is an adaptive network coding method is proposed for disruptive network. In MORE, relays opportunistically form multiple paths on which packets are re-encoded and forwarded. Later work such as CodeMP[38] further studies adaptive network coding based on measured loss rates for TCP sessions. On the other hand, many existing adaptive network coding works in DTNs focus on applying network coding to reduce the number of transmissions of epidemic routing or in conjunction with probabilistic forwarding. In [39], Y. Lin et al. studies the tradeoff between performance and resource consumption in DTN and proposes to spread slightly more-than-needed number of coded packets to reduce the number of transmissions. In [40], M. Chuah et al. proposes CANCO, which spreads coded packets to only some of the nodes encountered by delivery predictability and friendliness metrics.

In this study, we implement and examine network coding performance in disruptive content-based network platform.

CHAPTER 3

ICAN: An Information-Centric Ad-Hoc Networking Architecture

In this chapter, we give an overview of the ICAN system architecture. We start from a discussion of the system requirements, and then proceed to introduce the representation and definition of network entities and context, the APIs provided by the system, and finally the general functionality on routing and caching of the system.

3.1 Requirements

We first discuss the key requirements of an information-centric VANET architecture in the following.

1. **Context-aware operations**
   It is well-known that the challenges in VANETs mainly come from two aspects: mobility and error-prone wireless channel; both lead to intermittent connectivity and unstable routes. The solutions are best made with the knowledge of context. For example, network condition-related context is helpful and necessary in ad-hoc routing decisions: an efficient multi-hop routing is the target solution in a dense urban scenario (say, a sudden power outage in New York City). In contrast, in a sparse rural network, an efficient DTN routing protocol [41][29][42] is the desired solution. On the other hand, application-related context is important for efficient ICN caching. For

example, it is not useful to keep a live video packet in the cache for a day. The ICN concept enables the possibility of embedding context awareness in packet processing and routing by association of packet names and application-related contexts. Therefore, we choose ICN as the core design of our system.

2. **Push-based and pull-based transport support**

   The current ICN model adopts a pull-based approach in which a receiver-oriented interest packet is required to transmit a corresponding data packet. Most traditional Internet applications fit this communication style, e.g. web browsing through roadside or mobile access points. However, many applications that are important in ad-hoc networks, such as private messaging and emergent notification, are push-based. The data of these applications must be generated and delivered in real-time. Although it is possible to realize sender-driven applications with the pull-based ICN model [43][44], these proposals require the interests be registered periodically in advance. However, the periodic registration may backfire in ad-hoc network as it creates large overhead under high mobility. Therefore, it is necessary to include push-based transport.

3. **Disruption-tolerant delivery support**

   It is common that in ad-hoc network that the scenarios are intermittent. Currently, the ICN model assumes the interest must be sent in an end-to-end fashion to retrieve the data. However, in cases where the network is mostly partitioned, the data objects must rely on a carry-and-forward delivery. It is important that the system also supports disruption-tolerant networking. Of course, the application must be delay-tolerant to leverage the DTN transport.

4. **Extendability**

In order to embed context awareness in the networking architecture, it is necessary to define a representation of context. One lessen we learned from the history of Internet is that we cannot predict all demands for the future. Therefore, it is important to preserve extendability in the context representation design.

5. **Fast deployability**

   ICAN aims at enabling fast deployment in various scenarios on regular mobile device. While it is possible to obtain hardware support, a MANET or VANET by itself must adapt to the current scenario and should be quickly-configurable. Therefore, we assume ICAN will be implemented as an overlay in the form of software applications for the ease of deployment and upgrades.

3.2 Context awareness foundation

3.2.1 Network entity representation

The context awareness depends on the naming of network entities. With a universal network entity naming, we are able to identify the context of each chunk, or packet, by mapping its associated name or target destination(s) to an application-related or network condition-related context in the core of networking platform. In ICAN, data, nodes, and geo-locations are all identifiable. The following explains how each is identified:

1. **Data**: Following the hierarchical naming in [7], each data chunk is uniquely identified and associated with data object it belongs to. We enforce the following data chunk naming format: *application_id/data_object_id/chunk_id*. Each application has a globally unique ID. Data object ID is defined by the application and must be unique in the application's object namespace. Chunk ID is the sequence number of a data chunk within the data object

and is automatically generated when the chunk (i.e. packets) is created.

2. **Node**: Each node has a unique node ID. For compatibility, we assume IP or MAC address can be used to identify a node.

3. **Geo-location**: Considering the applications requiring geocast ability, we promote the geo-location as a named entity. The naming of geo-location is simply assumed to be the GPS coordinates of a location plus a diameter.

3.2.2 Context and metadata

We categorize the context used by ICAN as *application-related context* and *network condition-related context*. The application-related context is represented as the metadata of a data object or an application. For extendability, we assume the metadata format can be configured using XML [45]. Note that the metadata is not sent in XML format. Instead, XML is used for software configuration to define the meaning of received encoded context.

Attribute	**Required**	**Value**	**Default Value**
Content type	Yes	Offline/Real-time	Offline
Effective Time	Yes	[0ms, ∞]	∞
Publicity	Yes	Public/Private	Private
Popularity	No	High/Medium/Low	N/A
Max delay	No	[0ms, ∞]	N/A

Table 3.1: Application Metadata Format

Applications are responsible for providing part of metadata of its data objects. The metadata format implemented is shown in Table 3.1, and a list of sample definitions are in Table 3.2. Three attributes are required: content type, effective time, and publicity. The content type indicates whether the application content is created in real time. Applications also indicate the effective time of its data. For example, a news website's homepage may have an effective time of 4 hours, as shown in table 3.2. Another required attribute is publicity. We define a data object as private if it is access-restricted (e.g. encrypted data such as Facebook notification). If the application does not provide metadata, default values are used. Optional metadata fields may be indicated by the application designer or generated by the system automatically. In our example, the maximum delay is indicated by the application, and the popularity is collected based on the statistics of request frequency.

Application-related metadata can be disseminated periodically or on-demand. We define a default metadata dissemination service for answering on-demand requests. In general, the application-related metadata is retrieved on-demand for most network entities. Only the emergent applications or service may need periodic application-related metadata dissemination.

ICAN generates the network condition-related context automatically. Nodes maintain their own context including its location, a list of known connected nodes, and a list of out-of-contact nodes. We assume ICAN nodes use localization methods such as GPS to determine their locations; the list of known connected nodes is maintained by recording overheard network traffic; the list of out-of-contact nodes is maintained by implicitly detecting the retransmission failures towards known destination.

The location of a node is embedded in every packet forwarded by the node. All nodes collect the location of its neighbors, from both processed and overheard data chunks. In this way, ICAN implicitly collects the network condition-related context necessary for processing and adapts suitable forwarding and caching strategies accordingly. The complete network condition-related metadata are not disseminated by default, but is available upon request.

App or Data Object Name	**Context type**	**Effective time**	**Public-ity**	**Popula-rity**	**Max de-lay**
Facebook	Offline	∞	Private	N/A	
Car-A/ break-alert	Real-time	1 min	Public	High	2ms
Map/ Westwood-blvd	Offline	∞	Public	High	N/A
Traffic/ Westwood-blvd	Real-time	15mins	Public	High	N/A
NYTimes/ homepage	Offline	4hrs	Public	Medium	N/A

Table 3.2: Sample metadata

3.3 System Overview

3.3.1 Application API

The ICAN APIs provided for user applications are as follows.

1. *query(data_object_name, valid_time)*: for the requester to retrieve/subscribe to the specified data object. The query is valid for the period of time specified.

2. *put(data_object_name, data_object_content , metadata)*: for the data provider to publish a data object with a certain data name with a the associated

metadata. If metadata is not specified, the default value of the associated application is assumed.

3. *set(data_object_name, metadata)*: for the provider to specify the metadata of a data object. If metadata is not specified, the default value is assumed.

4. *set(application_name, metadata)*: for the provider to specify the metadata of an application.

5. *push(data_object_name, data_object_content, metadata, destination_names)*: used to push a data object to a certain destination. The destination is either a set of nodes or a geo-location. With this API, any push-based application using unicast, multicast, or geocast can be implemented.

3.3.2 System architecture

Figure 3.1 illustrates a typical ICAN node. The application layer publishes data objects and provides metadata to ICAN by the application APIs. The data objects then will be fragmented to chunks, named accordingly, and saved to the content store.

ICAN is an all-broadcast system. At the bottom, the broadcast layer ensures reliable hop-by-hop broadcast-based transmission (see 3.6.2). The data networking layer extends the current ICN architecture with context, push transport, and DTN mode support. It consists of four basic components: context plane, content storage plane, traffic control/aggregation plane, and routing plane. The context plane takes care of the collection, resolution, and storage of the context. It is responsible to parse received packets, collect network condition-related context, estimate network conditions such as capacity and loss rates, maintain neighbour lists and out-of-contact node lists, and records application and data object metadata. In addition, it resolves the context for a given packet and provides information needed by the context-aware caching, routing, and traffic engineering decisions.

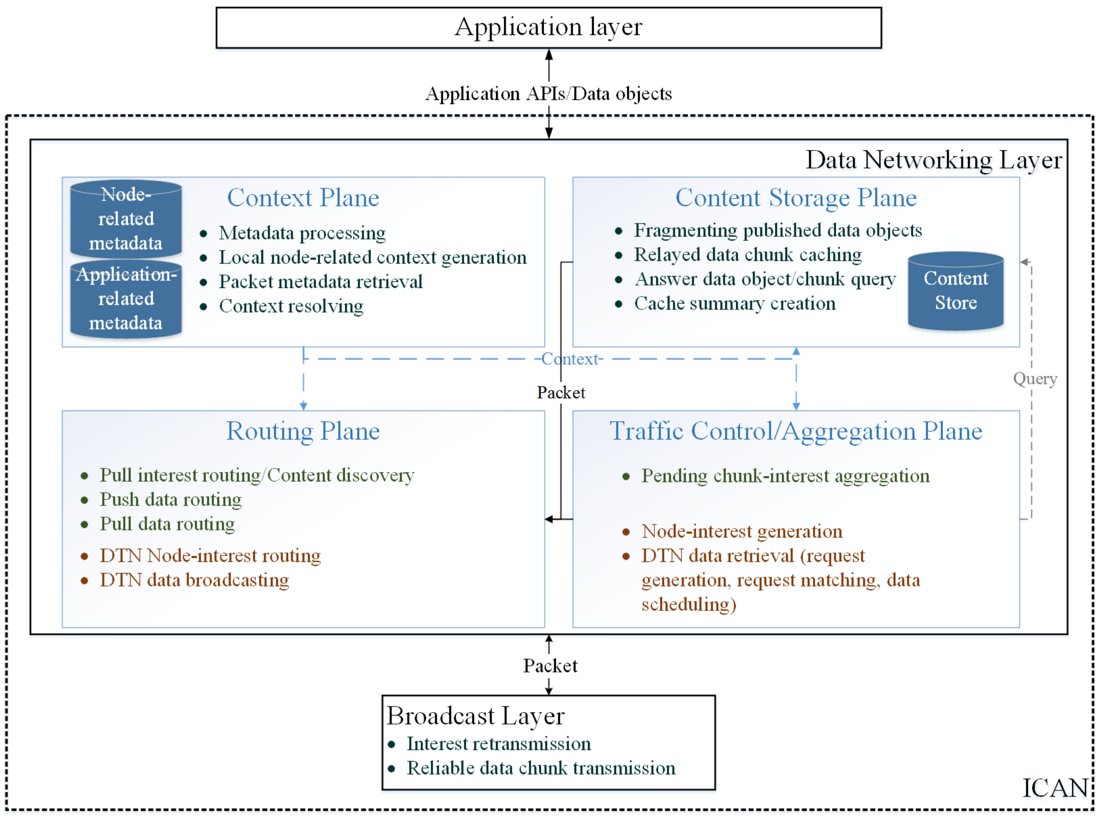
Application layer
Application APIs/Data objects
Data Networking Layer
Node-related metadata
Application-related metadata
Context Plane
Metadata processing
Local node-related context generation
Packet metadata retrieval
Context resolving
Content Storage Plane
Fragmenting published data objects
Relayed data chunk caching
Answer data object/chunk query
Cache summary creation
Content Store
Context
Packet
Query
Routing Plane
Pull interest routing/Content discovery
Push data routing
Pull data routing
DTN Node-interest routing
DTN data broadcasting
Traffic Control/Aggregation Plane
Pending chunk-interest aggregation
Node-interest generation
DTN data retrieval (request generation, request matching, data scheduling)
Packet
Broadcast Layer
Interest retransmission
Reliable data chunk transmission
ICAN

Figure 3.1: ICAN System Architecture

The content storage plane consists of the content store and is responsible for fragmenting the locally published data objects, and opportunistically advertising the cache summary of delay-tolerant data objects. The traffic control plane handles the traffic engineering and packet aggregation to better utilize the precious capacity of the wireless channel. Unless the data source is judged out of contact, ICAN follows the multi-hop communication model of ICN for pull-based data retrieval. In this case, we keep the ICN PIT in [7] to aggregate interests for the same data chunks from different nodes. This reduces the traffic load as the data follows the breadcrumb of the interest. On the other hand, if the destination is detected unreachable, the packet is delivered in DTN mode (3.4.2). DTN packets are deferred, compiled at the node level and an epidemic style data retrieval is pursued. The routing plane is in charge of packet routing, that is, when and if a packet is re-broadcast. Based on the application-related and network condition-related context, the packets are categorized as one of the following: pull interest, pull data, push data, DTN request, or DTN data. Each is handled by a dedicated routing engine. Each routing engine is configured with a corresponding forwarding protocol. Note that as the context format is extendable, different algorithms can be easily implemented as plug-ins to extend any of the routing engines.

3.3.3 System service

ICAN considers common data query services such as metadata dissemination and network entity discovery services as a special type of application. These services are installed on every node. Unlike the user applications, services use pre-defined service name prefixes concatenated by the node or geo-location identifier to indicate the information being queried. For example, a node location query of a location service is named *location_service/node_ID*

ICAN allows services to be installed at system level, and thus the service requests can be answered by any node that has the specified service installed, leveraging the context and caching functionalities. This means services such as node location or content discovery are distributed, better suited the ad-hoc environment.

3.4 Packet Forwarding

ICAN includes two packet networking paradigms. By default, the destination (data source or target node) is assumed reachable and the end-to-end multi-hop forwarding approach is pursued. If the destination is known to be or is judged unreachable (i.e. by timeout), and if the application is indicated delay-tolerant, the packet is processed in DTN mode. In the following, we summarize the packet processing procedure for these two modes.

3.4.1 Packet processing: multi-hop forwarding mode

When the data communication is initiated by the receiver, a *pull-interest*[1] must be sent to retrieve a data chunk. We summarize the pull-based packet processing procedure as follows. Incoming interest packets from application, service, or broadcast layer first enter the context plane. After inspecting the packets, updated metadata is stored in the context knowledge base. Next, ICAN searches the content store to find matching data. If nothing is found, the interest name prefix is matched to applications and installed services. If the prefix matches an application, ICAN queries the application with the data object name to get the on-demand data object, fragments it to chunks, and sends the requested chunk out to the requesting interface. If the prefix matches a service's, the distributed service may or may not reply to the interest after inspecting it. If the service decides not to answer the query, the interest packet is returned to ICAN for further processing. At this point, the node searches its PIT to decide if it has previously sent a interest with the same name. Pull-interests from different requesters are integrated to avoid redundant transmissions, which means the relay does not transmit the interest packet again but only adds the requesting face to the existing PIT entry. Otherwise, the interest to be forwarded is handed to the interest routing module. Based on the context, the interest routing module decides whether to broadcast the packet right away. If the interest is not consumed by the corresponding data after several retries at the requester node, ICAN checks the corresponding application's context and if it is delay-tolerant, the interest is then turned to DTN mode.

Pull-data packet processing is as follows. A pull-data packet along with metadata it carries are inspected by the context generator. The data content is then cached by CS. At this point, CS consults the context resolver for caching decisions. Later, if the data packet is not targeted to a local service or application,

[1]In the later text, we use pull-interest and interest interchangeably.

it is handed to the pull-data routing module, which lets the packet follow the breadcrumb path of the interest.

When the data delivery is initiated by the sender, the data packet is processed as a push-data. Push-data packet processing is similar to pull-data packet processing. The major difference is the routing module: the push-data is handed to the push-data routing, and a multi-hop routing method using node or geo-location as final destination of the packet will be used to deliver the packet to its destination.

3.4.2 Packet processing: disruption tolerant mode

The disruption-tolerant mode of ICAN (DT-ICAN)[2] subsumes both the family of peer-to-peer content dissemination network (e.g. Haggle[46]) in which interests are propagated in an epidemic fashion as well as the family of ICN [7] in which data is cached as uniquely identifiable chunks. This leaves us higher data availability in disruptive ad-hoc network and in the mean time preserves the possibility of content-aware caching/routing design. Note that we focus on the epidemic data requesting in DT-ICAN, as the applications that require push-based transport are usually emergent and not delay-tolerant.

DT-ICAN reduces the bandwidth consumption by per-node interest aggregation. This is done by splitting the functionality of interest to two periodically broadcast control messages: the *node-interest*, which indicates the data objects a node wants, and the *request*, which represents the data object the node is currently trying to retrieve within one hop. In addition, each node periodically advertises its *cache summary* to assist efficient content requesting. Node-interest, request, and cache summary are all represented by Bloomfilters [47].

1. Node-Interest

   To improve the scalability in disruptive networks, nodes do not request data

[2]https://github.com/uclanrl/dt-icansim

chunks by chunk-based pull-interests. Each node aggregates the data object requests from its applications as a node-interest. The node-interest is compiled in file object basis. In other words, a node indicates the *file object IDs* instead of chunk IDs to speed up the retrieval. Instead of instantly flooding node-interests, nodes opportunistically and periodically propagate the node-interests received from others over multiple hops when the bandwidth is sufficient. (6.1.2)

2. Request

   As the encounter intervals in a sparse network can be very limited, the node-interests do not trigger data transmission immediately because otherwise the amount of data transfer triggered can be out of control. We introduce the *request* message, which identifies a subset of data objects that a node is willing to receive at the present time. Note that a request may consist of not only data objects the node itself subscribes but also the data objects others are interested in.

   Requests are broadcast only within one hop to retrieve data from neighbours. When a new request comes in, a node examines the request with the names of data in CS, compiles a list of data objects to offer, and then initiates data chunk transmissions in order using the handshake procedure (6.1). The request transmission may be triggered when new contact is discovered or periodically. Note that the node has the freedom to decide what contents to pull based on the volume of node-interests it receive, the network condition, and its local content prioritization policy. We assume nodes decide the amount of data to retrieve from the newly-encountered neighbor based on the available bandwidth 6.1.3, and aggregates the retrieval in one request to prevent overwhelming data transmissions from multiple caches.

3. Cache summary

   Each node periodically generates its own *cache summary* ane broadcast it to one-hop neighbors. The cache summary includes the data object IDs of fully cached file and chunk IDs of partially cached ones. The cache summaries are leveraged by nodes to prioritize which data chunks to send and request. Without cache summaries, nodes may blindly pull redundant data based on previously received node-interests. All nodes also update neighbors' cache summaries by the data object names carried in the control messages and data overheard.

More details of DT-ICAN operations will be described in 6.1.

3.5 All-broadcast routing

3.5.1 Basic components of routing plane

In order to better utilize the caching capability of ICN, we design ICAN as an all broadcast networking platform. We hereby describe the basic routing principles of the routing engines.

1. **Name-based multi-hop packet propagation**
   As the pull data is forwarded via breadcrumb path of pull-interest, the multi-hop delivery path of pull-data depends on the interest routing which delivers the pull-interest packets by data chunk name. On the other hand, the push-data will be forwarded based on the node or geo-location ID. Fortunately, both can be done using the same name-based routing protocols, as all destinations are named in ICAN. The real challenges of routing and forwarding come from the lossy channel and high mobility. Therefore, we adopt the opportunistic routing concept in ICAN. For multi-hop delivery, ICAN floods the packets when the destination location is unknown, but directs the packets to known locations via one or more opportunistically constructed paths otherwise. In 3.5.2 and 3.5.3, we will introduce the concepts of our ap-

plied mobility-resistent name-based flooding and the proposed geo-assisted opportunistic routing, respectively. All routing protocols proposed in the following chapters are variants of the two concepts. For example, an extended geo-assisted opportunistic routing protocol aiming at urban VANET will be illustrated later in chapter 4.

2. **DTN routing**

   When the target destination is unreachable, a dissemination of the node-interest is needed. Most dissemination-based protocols in such environments are variations of Epidemic Routing (ER) [29], which diffuses messages into networks by the cache-and-forward technique. The overhead of epidemic propagation can be aggressively reduced by techniques in which a node selects to which encounter nodes a message is forwarded [31][41][42]; many of these methods utilize mobility patterns or contact history to further reduce the unnecessary propagation. While these techniques can be easily integrated in ICAN as new DTN interest routing plugins, we note that the study of epidemic dissemination is beyond the scope of this study and we assume a simple epidemic dissemination of DTN interests in our experiments.

3.5.2 Low cost mobility-resistant flooding

In many situations such as unknown location content search and limited-scope data pushing, flooding-like packet propagation is necessary. The simplest form of flooding in an all-broadcast system is to let all nodes rebroadcast the packet as long as the nodes are within the specified flooding range (i.e. restricted by hop counts, or defined geographic area). However, this approach is costly as many redundant packets broadcast will occur in a dense scenario. There have been abundant research on the flooding issue in wireless environment; most involve computing a Multi-Point Relay (MPR) [26] to select the proper forwarding nodes achieving lowest cost. However, it is well-known that in a highly mobile network such as vehicular network, the topology changes so rapidly that it is very difficult to compute an optimal MPR. Therefore, instead of computing MPR, we apply a timer-based Smart Flooding (SF) mechanism.

SF utilizes the timer-based rebroadcast concept [8]. The basic idea is as follows: when a node receives a packet, it first decides if it is an eligible forwarder of the packet. Afterwards, the eligible forwarders contend for the opportunity of rebroadcast. Each eligible forwarder sets a rebroadcast timer based on a function of its distance to destination. Upon the expiration of its rebroadcast timer, the forwarder sends the packet. Before the timer expires, if the node hears other nodes rebroadcast the same packet, it may cancel its scheduled transmission. Hence, among all the nodes within a small range, only the highest-priority forwarder will rebroadcast the packet first due to its shorter timer.

In SF, all nodes receiving a packet are eligible forwarders and they all join the contention. The primary goal is to expand the propagation range. In other words, the farther neighbors must be given higher priority. Therefore, the SF expiration timer of an eligible forwarder is calculated as follows.

$$T_f = T_{dist}\frac{(D_{max} - min(D_{max}, D_{transmitter}))}{D_{max}} \tag{3.1}$$

T_{dist} is the defined maximum waiting time, D_{max} is the node's maximum transmission range, $D_{transmitter}$ is the distance from the node to the last hop who it first receives the packet from. Note that the farther this node is from the last hop, the shorter it needs to wait for rebroadcasting the packet.

3.5.3 Geo-assisted opportunistic routing

A reliable single path routing mechanism is needed when a packet has a target destination. In ICAN, we associate all named destinations to their geographic coordinates and design a geo-assisted Smart Single-Path forwarding (SSP) for

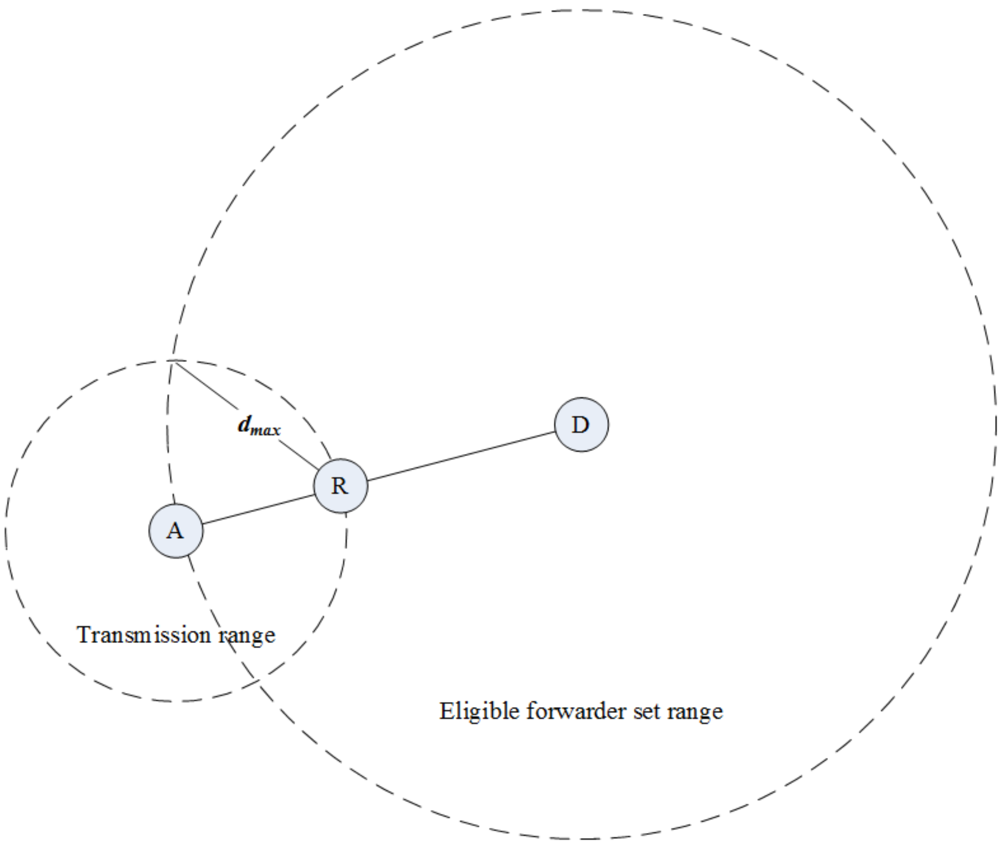

Figure 3.2: Geo-Assisted SSP distance estimation

the purpose. SSP also utilizes the timer-based rebroadcast mechanism described above. Unlike traditional routing algorithm in which a next hop is deterministically specified, SSP is resilient to rapid topology changes with minimal control overhead.

In SSP, nodes closer to the destination are preferred. We define relay nodes who are closer to the destination carried in the packet than the last hop is are eligible forwarders and only the eligible forwarders will join the contention. The ineligible nodes automatically drop the received packets. The eligible forwarders are prioritized by their normalized distance to the destination; thus, their rebroadcast timer is calculated as follows:

$$T_s = T_{dist} \frac{d_{ref}}{d_{max}} \tag{3.2}$$

d_{ref} is the distance from the node to the reference point and d_max is the maximum distance between the eligible forwarders within last hop's range to the reference point. The reference point (position R in Figure 3.2) is used to normalize the distance to destination of all eligible forwarders and is defined as the closest possible geo-coordinate to the destination within last hop's transmission range. In Figure 3.2, A and D represents the last hop and the destination, respectively.

3.6 Broadcast layer

3.6.1 Data storm control

The all-broadcast system incurs an issue particularly in ad-hoc network. Since multiple caches of the same data chunk may exist in a neighborhood, it is likely a broadcast interest may lead to a data storm, meaning many caches try to broadcast the corresponding data at the same time. To migrate the issue, for pull-data, all cache nodes randomly select a short backoff time, and the data transmissions are suppressed if the cache hears another data transmission before the timer expires.

3.6.2 Reliable broadcast

All transmissions in ICAN are done by MAC layer broadcast to better leverage the wireless broadcast channel. The consequence of this design is that there is no support from MAC layer retransmission. Therefore, we include a reliable broadcast mechanism to guarantee robust transmission with minimum retransmissions.

The reliable broadcast utilizes the packet names a node heard to ensure delivery. To avoid severe congestion introduced by duplicate packets, all packets carry nonce in our system. Packets are retransmitted, up to a certain limit of times, if a relay node does not detect a progress is made. A node judges there is a progress made for a particular packet if one of the following conditions is met:

1. A packet carrying the same nonce is received from neighbors

2. The node receives an ACK that acknowledges a nonce carried by a data packet. Note that this implies that when a packet reaches its destination, the destination must send an ACK with the received packet nonce to stop the retransmission from the last hop.

3. The node receives a data packet carrying the same name as that of the previously received interest.

CHAPTER 4

Exploring The Parameter Space In Urban VANET

It is well known that in urban VANET, the topology changes rapidly, and thus route maintenance and discovery of traditional IP networking are costly. Therefore, it is natural to believe that the pervasive caching and multi-path nature of ICN are likely to be beneficial, that is, to provide better content availability under ICN paradigm. However, the tradeoff between the content availability gain and the incurred overhead of different design options in routing and caching require an in-depth study. In this and the following chapter, we concentrate on discussing the technical challenges of urban VANET use case. We evaluate the contribution of possible caching and routing design options by extensive simulation through synthetic application and vehicle traffic traces. To our knowledge, this is the first study carrying out such simulation even in the IP-based VANET field. Our results illustrate the performance gain of an ICN-like VANET over IP-based, peer-to-peer VANET.

4.1 Design Options

4.1.1 Caching

We study three general caching options for urban VANET. Namely, **pervasive caching (ALL)**, **peer-only caching (P2P)**, and **no caching (NONE)**.

- Pervasive caching (ALL): pervasive caching is the standard assumption of ICNs; it assumes that all nodes in the network will cache all data chunks they received. When the cache space is insufficient, the new chunk is accepted and one of the old chunk must be ejected.

- Peer-only caching (P2P): peer-only caching is considered the state-of-art of VANET caching. While VANET applications are still in its infancy, many VANET applications assume the system is built as an overlay P2P network [48] in which the participating peers carry and forward the information published by its peers. In this case, only the peers interested in the same application files may cache the file. We consider this option a baseline of VANET caching paradigm.

- No caching (NONE): we refer to the system without caching capability as no caching. In other words, no caching represents a pure host-based approach in which the contents are simply forwarded but not replicated.

As one may expect, the more nodes caching the same information the higher availability of the information will be. While this conclusion comes in natural, we are interested in investigating the performance gain of pervasive caching over the other two approaches in the mobile ad-hoc network, as there is a tradeoff between the complexity of cached content management, the required cache size, and the content availability. We expect the results will shed the light and provide insight on whether a pervasive caching is worthy of the processing and space overhead incurred.

4.1.2 Routing

We study two routing design options for urban VANET. Note that here we assume a reliable route can be found and therefore we focus only on the destination selection. The two potential options are **nearest replica routing (NR)** and

source-only (i.e. origin-only) routing (SR), as defined in [49].

- Nearest replica routing (NR) is the default behavior assumed in current ICN systems [7][49]. The purpose of nearest replica routing is to take full advantage of the data cached at the routers. In nearest replica routing, the interest is sent to the cache closest to the requester.
- Source-only routing (SR) means that the interest is directed towards the original source (publisher). It is still possible that a cache on the path between the data source and the requester may be utilized, but the cache-hit probability may be lower and the response time may be longer if the nearest cache has moved to a location off the path from requester to origin.

In both approaches, a path is constructed between the requester and the destination of the interest. Note that although the process to locate a content by name of ICN, referred as content discovery, is an important issue, our focus in this chapter is to study the best achievable performance of each design option. Hence, we assume there is an efficient way to locate data object with constant or near-negligible overhead. In reality, to achieve nearest replica routing in VANET, the locations of replicas for each piece of content must be recorded so that one can ensure the nearest replica is targeted, requiring high maintenance overhead. Therefore, it is important to evaluate the performance gain of nearest replica routing so that one can gain the insight for content discovery design. For this purpose, in the simulation in this chapter, the locations of nearest caches and data sources are obtained without cost. Note that our conclusion will only be reinforced if the content discovery cost is considered.

4.1.3 Problem statement

[49] looked into the design space of ICN in the wired Internet and pointed out that the benefits of ICN's ubiquitous caching and nearest replica routing are limited

and can mostly be obtained by incremental implementations. We expect the situation will be different especially in sparse and disruptive VANET, as end to end connectivity cannot be maintained and pervasive caching is a must for carry-and-forward. However, unlike in disruptive MANET/VANET, in urban VANET, the content are in general highly reachable despite the frequently changing routes, and thus it is necessary to perform an extensive study considering the vehicle movement and the application patterns to best understand the routing and caching design options and their combined effects in the urban VANET environment.

Inspired by [49], we study the intersection of the ICN paradigm and vehicle-to-vehicle communication in this chapter. We aim at answering two questions:

1. How much is the gain of ICN pervasive caching compared to the existing peer-only caching paradigm in VANET?
2. Is it necessary to monitor the locations of all replica? In other words, does nearest-replica routing achieve significant gain over source-only routing?

4.2 Simulation settings

The simulation is conducted using QualNet 6.1. Since we focus on exploring the general VANET ICN design space, our simulation is based on an abstraction of non-DTN ICAN, in which the data is chopped to chunks and each chunk has a unique name that associates to its parent data object, the system is all-broadcast to better explore the cached contents at relays or peers, and the data follows the breadcrumb path of interests. All simulations are run 1000 seconds. The applications start at 400 seconds and continue being active for 300 seconds. The detailed settings are described in the following.

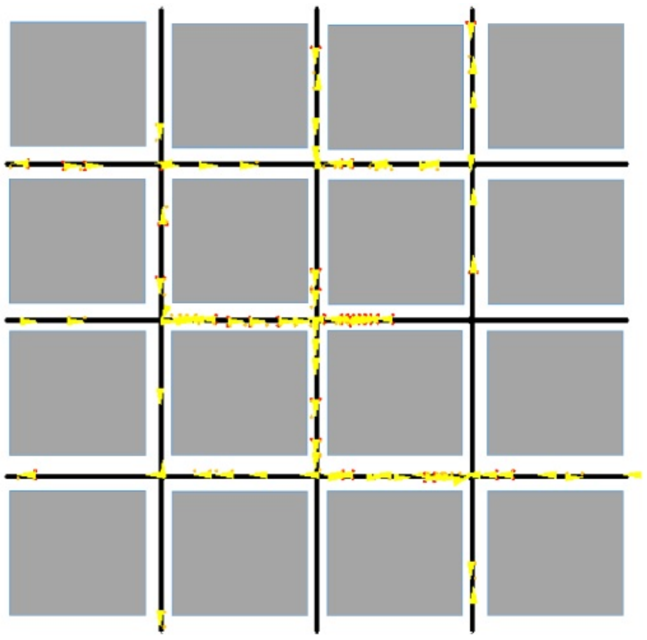

Figure 4.1: Simulation map

4.2.1 Mobility and propagation model

We simulate an urban VANET scenario using a $2km^2$ Washington DC map from Tiger/Line file [50], as shown in figure 4.1. Each road segment is 500 meters long, bidirectional, with two lanes in each direction. The gray areas represent the obstacles.

The vehicle movements are simulated using SUMO [51]. SUMO is an open source road traffic simulation package designed for large road networks. We generated three traces for 400 vehicles using different random seeds. Figure 4.1 shows a snapshot of one of the generated trace. The yellow points represent the vehicles.

We use CORNER[52] as the radio propagation model. CORNER is a channel model for urban VANETs. It takes the street map as the input and generates path loss using a function of the relative positions of two nodes. CORNER classifies pairs of vehicles into three cases: line of sight (LOS), non line of sight with one intersection along the path (NLOS1) and non line of sight with two intersections along the path (NLOS2). For each case, the path loss is calculated differently.

4.2.2 ICN Topology-Assisted Geo-Opportunistic forwarding (ICN-TAGO)

Although with built-in chunk-level caching, VANET ICN still faces the challenges incurred by the intermittent links and unstable capacity caused by obstacles and mobility, which is well simulated by the CORNER model. Therefore, one critical aspect for the system is to construct a robust routing mechanism conquering such environments. To handle the obstacles (buildings) in urban VANET. We extend SSP (3.5.3) and propose ICN Topology-Assisted Geo-Opportunistic forwarding (ICN-TAGO) with the road topology awareness. The idea is summarized as follows.

1. The requester first retrieves the destination location (a geo-coordinates of the target data holder) from the content location and discovery service. It then stamps the interest packet with its own location, the destination location, and then broadcast the packet. We will discuss the content discovery methods in more details later in chapter 5. Since the purpose of this chapter is to explore the ICN design space generally, we simply assume the location is known with negligible overhead instead of selecting a particular content discovery mechanism.

2. Upon receiving a request, a relay node first decides whether it is an eligible forwarder by calculating the possible routes from the last hop to the destination. Using the destination location, its own current location, and the knowledge of the area map, a node can calculate a virtual route represented by the intersections of the map by basic Dijkstra's algorithm. A node judges itself to be an eligible forwarder if it is on one of the virtual path between last hop and the destination. Otherwise it drops the packet.

3. Eligible forwarders then enter the contention phase. During contention, a relay node sets a timer based on its computed virtual path to the destination and the last hop location. The relay re-broadcasts the packet if the timer expires before it hears a re-broadcast from an eligible forwarder who has higher priority. Otherwise, it drops the packet.

4. The process repeats until the interest reaches a data holder or the recorded destination location. Once the data holder receives the interest, it broadcasts the data. Upon receiving a data, the relays cancel their timer and drop the request packets. The data then follows the breadcrumb path back to the requester.

The pseudocode of the contention timer algorithm is presented in algorithm 1. The forwarders are prioritized as follows. Nodes calculate the target intersection of the last hop based on the last hop location carried by the request and their own target intersections. Note that the target intersection can be easily computed using any shortest path algorithm with the road map defined as a set of links between intersection locations, and the location of destination and a given node. The map information making the topology awareness possible is commonly available in navigation system. In our implementation, we use Dijkstra's algorithm to compute the target intersection.

If a node is closer to the destination than the target intersection of the last hop, it is in the group of high priority class. If it resides between the last hop and the target intersection of the last hop, meaning it is on the same road segment as the last hop's, it is in the group of low priority class. In all other cases, it is an ineligible forwarder. The high priority class nodes set their timer by

$$T = T_{high} \frac{d(loc_m, target_m)}{TxRange} \tag{4.1}$$

Algorithm 1 ICN-TAGO Set Expiration Timer

```
1: procedure SetTimer($map, last, dest, req$)
2:     if I am requester then
3:         $timer \leftarrow 0$, return $timer$
4:     else if I am dataholder then
5:         drop $req$ and contend for data broadcast, return
6:     else
7:         $eligible \leftarrow false$,$updatedtarget \leftarrow false$
8:         $target_l \leftarrow NextIntersection(map, last, dest)$
9:         if I am between $target_l$ and $last$ then
10:            $eligible \leftarrow true$
11:        else if I am on line-of-sight to $target_l$ then
12:            $loc \leftarrow$ my position, $target_m \leftarrow NextIntersection(map, loc, dest)$
13:            if $target_m$ is closer to $dest$ than $target_l$ then
14:                $eligible \leftarrow true$, $updatedtarget \leftarrow true$
15:            end if
16:        end if
17:        if $eligible = true$ then
18:            if $updatedtarget = true$ then
19:                $timer \leftarrow equation(4.1)$
20:            else
21:                $timer \leftarrow equation(4.2)$
22:            end if
23:            return $timer$
24:        else
25:            drop $req$, return
26:        end if
27:    end if
28: end procedure
```

Algorithm 2 ICN-TAGO Cancelling Mechanism

```
1: procedure CANCEL($map, last, dest, packet$)
2:     if $packet$ is the corresponding data packet then
3:         cancel transmission
4:         return
5:     else if $packet$ is the request then
6:         $loc \leftarrow$ my position
7:         $target_m \leftarrow NextIntersection(map, loc, dest)$
8:         $target_l \leftarrow NextIntersection(map, last, dest)$
9:         if $target_m \neq target_l$ then
10:            cancel transmission
11:            return
12:        else if $last$ is closer to $target_m$ then
13:            cancel transmission
14:            return
15:        end if
16:    end if
17:    return
18: end procedure
```

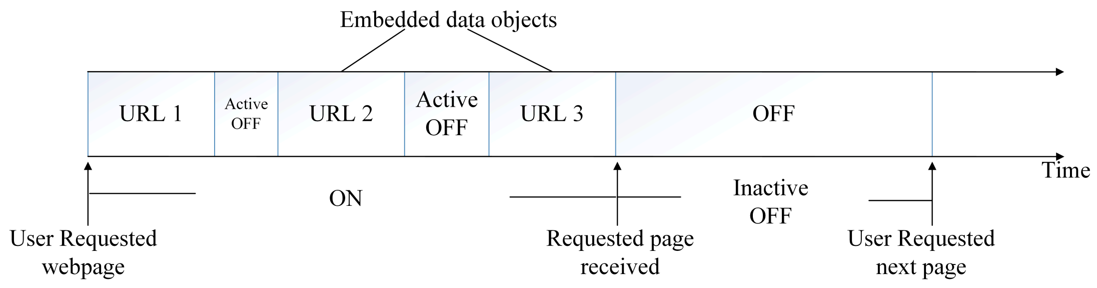

Figure 4.2: On/Off User Behavior

where T_{high} is the maximum expiration time for high priority class, loc_m is the location of the eligible forwarder m, $target_m$ is the target intersection of forwarder m, and $d(i, j)$ represents the distance between location i and j. Note that for high priority class, $target_m$ may be different from the target intersection of last hop, and this new target is closer to the destination. By definition, it is possible that the requests are propagated via multiple paths (towards multiple target intersections), but the propagation is directional. This multi-path exploration helps reduce the chance that the request encounters a "hole".

For low priority class, the target intersection is the same as the last hop's, and the expiration timer is set in a range of $[T_{high}, T_{low}]$ by

$$T = T_{high} + (T_{low} - T_{high})\frac{d(loc_m, target_m)}{TxRange} \tag{4.2}$$

where T_{low} is the maximum expiration time for low priority class.

The timer cancelling mechanism in algorithm 2.

4.2.3 Web access user behavior model

The measured performance changes dramatically with different application traffic in mobile ICN due to the fact that contents are cached on a dynamic path between mobile data holders and requesters. To obtain insights for a more realistic application traffic, we simulate the user behavior based on the SURGE model [53]. Each requester issues requests new data objects following an On/Off model [54].

The model is represented as a state machine in figure 4.2. The On state represents the duration when an attempt to request a web page is made. Each web page is associated with a batch of data objects. The batch size follows a Pareto distribution. The data objects are retrieved one by one. When the batch transmission is complete, no matter it is successful or not, the application goes to Off state. This state represents the "user thinking time" and is Pareto distributed. The On state itself is another state machine including FileSend state and ActiveOff state. These two states represent the time the requests for a particular data object are being sent and the system processing time between requesting two data objects, respectively. The ActiveOff time follows Weibull distribution. As AcitveOff time is supposed to be much smaller than the inactive Off time, we cut off the Weibull distribution tail so that the maximum ActiveOff time is 1 second. As suggested by [53], we generate the data object size using the hybrid Lognormal/Pareto distribution with a 93% cutoff. The file popularity follows a Zipf-like distribution [55] with α set to 0.8. The parameters of all distributions are summarized in table 4.1.

Component	**Model**	**Probability Density Function**	**Parameters**
File size-body	Lognormal	$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-(lnx-\mu)^2/2\sigma^2}$	$\mu = 9.357;\sigma = 1.318$
File size-tail	Pareto	$p(x) = \alpha k^{\alpha} x^{-(\alpha+1)}$	$k = 133K;\alpha = 1.1$
Popularity	Zipf		
Temporal locality	Lognormal	$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-(lnx-\mu)^2/2\sigma^2}$	$\mu = 1.5;\sigma = 0.8$
Request sizes	Pareto	$p(x) = \alpha k^{\alpha} x^{-(\alpha+1)}$	$k = 1000;\alpha = 1.0$
Active OFF time	Weibull	$\frac{bx^{b-1}}{a^b} e^{-(x/a)^b}$	$a = 1.46;b = 0.382$
Inactive OFF time	Pareto	$p(x) = \alpha k^{\alpha} x^{-(\alpha+1)}$	$k = 1;\alpha = 1.5$
Embedded objects	Pareto	$p(x) = \alpha k^{\alpha} x^{-(\alpha+1)}$	$k = 1;\alpha = 2.43$

Table 4.1: Summary Statistics for models

To ensure the data object universe size is large enough for the simulation duration, we generate the data object universe with the size

$$U_{size} = (N_v/N_{neighbor})T_{sim}/T_{retrieve} \tag{4.3}$$

where N_v represents the total number of vehicles, T_{sim} is the duration of active simulation time, $T_{retrieve}$ is the expected time for an average web page retrieval. $N_{neighbor}$ is the number of cars in one hop distance estimated by

$$N_{neighbor} = \frac{TxRange}{W_v + D_{safe}} \tag{4.4}$$

where $TxRange$ is the maximum transmission range, W_v is the width of a vehicle, and $D_s afe$ is the safety distance between cars. A mid-size car has width about 4.8 meter, and the safety distance between two cars is 33 meters for 25 mph speed. The maximum transmission range is 210 meters for 54Mbps data rate.

The average web page retrieval time is estimated by

$$T_{retrieve} = \frac{N_{obj}N_{filesize}}{R} + T_{off} \tag{4.5}$$

where N_{obj} is the expected number of data objects in a batch, $N_{filesize}$ is the expected size of a data object, R is the data rate, T_{off} is the expected inactive off time. The numbers are obtained by the mean of the corresponding distributions.

In addition, we implement a sliding window-based flow control mechanism at the receiver side, similar to that in TCP, to ensure consecutive chunk interests. The sliding window size is a constant equivalent to the data chunk size.

4.2.4 Metrics

All metrics used in the experiments are listed as follows.

- Completion ratio: the fraction of data chunks successfully received by the data consumer out of all requested chunks.

- File retrieval ratio: the fraction of data objects (files) successfully received by the data consumer out of all initiated data object retrieval.
- Response time: the time interval from the moment a data consumer first sends an interest to the point when the data consumer receives the first corresponding data chunk.
- Cache hits: the number of times when a relay finds the corresponding data chunk in its cache.
- Goodput (Bps): the total data objects successfully received on the requester side in bytes each second.
- Aggregate traffic: the total number of bytes sent by the network over the entire simulation time.

4.3 Simulation results

4.3.1 Design options

We first compare the six combinations of design options: NR+ALL, SR+ALL, NR+P2P, SR+P2P, NR+NONE, and SR+NONE. While keeping the total number of nodes 400, we vary the number of requesters in this experiment between 10, 100, and 400, to explore how each option is affected by busier networks.

The results of completion ratio is shown in Figure 4.3. Comparing the three caching options, we find that the completion ratio of pervasive caching (ALL) improves from no caching's 50-55% and peer-only caching's 53-57% to 78-88%. This shows that pervasive caching is useful in the ad-hoc environment as the mobility is high. Interestingly, pervasive caching achieves 33% improvement on completion ratio over peer-only caching. Note that in peer-only caching, only peers requesting the same data objects will cache that particular data object, and

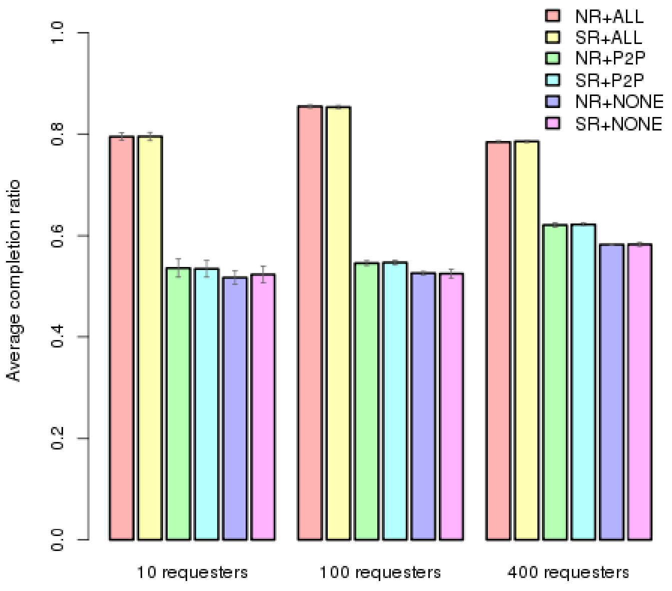

Figure 4.3: Design options: completion ratio

hence the number of data objects cached at each node is less than that in pervasive caching, even if the number of requesters is the same as the total number of nodes. This is a significant difference between the ad-hoc network and the Internet [49]. This result shows that pervasive caching is helpful in an ad-hoc network, even if the topology is not disruptive. Note that the difference in file retrieval ratio is even more significant (Figure 4.4, as the user will abort the entire data object transmission when a chunk of the object is not received after 4 requester-side reinitiation. We conclude that pervasive caching is critical to the performance in an urban VANET.

The response time result is shown in Figure 4.5. Surprisingly, examining the results for nearest-cache routing and source-only routing with the same caching paradigm, the response time of two routing options are similar. This means that nearest cache routing achieves only limited improvement even when the mobility is high, and suggests that it is reasonable to eliminate cache location monitoring

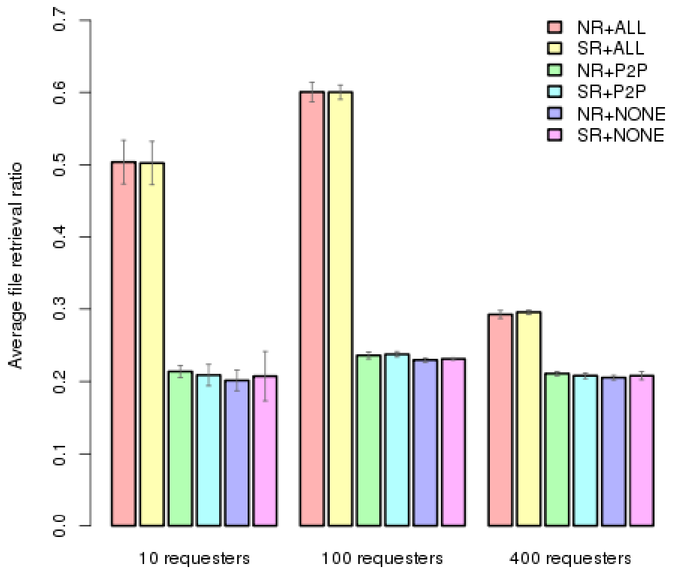

Figure 4.4: Design options: file retrieval ratio

to reduce overhead. The reason of this result is that no matter which caching mechanism is used, when a path is chosen, the content can only be cached on the path. As the movements of the vehicles must follow the road map, if the cache is reachable, it is likely that both nearest-replica routing and source-only routing can reach a data holder in similar distance when the interests from different requesters come in at different times.

We also observe that in Figure 4.5, the response time of pervasive caching is lower when the network load is moderate when number of requester is 10 or 100, as expected. However, when the number of requesters increase to equivalent to the total number of nodes, pervasive caching may cost higher response time. This is because pervasive caching incurs more redundant data transmissions despite the controlled data storm. The data collision is still likely to happen due to hidden-terminal and thus retransmissions may be frequent. However, considering the significant improvements in completion ratio and file retrieval ratio, one can

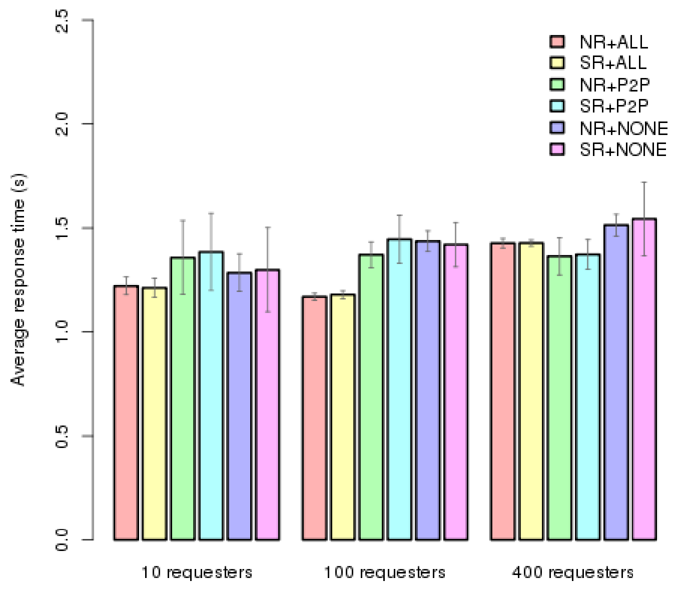

Figure 4.5: Design options: response time

judge that pervasive caching increases the data availability significantly and is a must in highly mobile ad-hoc network.

4.3.2 The effects of cache size

While pervasive caching brings significant performance improvements, the excessive storage overhead may be costly. Therefore, it is important to understand the tradeoff between cache size and performance. We performed a test varying cache size with 100 requesters. In this set of experiment, we use pervasive caching and source-only routing, and apply random cache replacement policy as it has been confirmed to achieve as good data availability as more sophisticated policies with pervasive caching [56].

As shown in figure 4.6, the completion ratio reaches the ceiling of 85% soon and remains flat after 6000KB. Looking at the response time trend in figure 4.7,

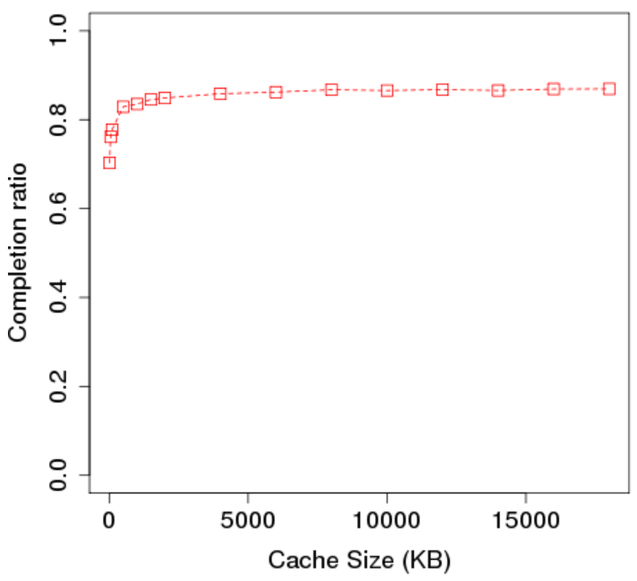

Figure 4.6: Cache size test: completion ratio

a sweet spot 6000KB is sufficient for this scenario. The reason why only a small cache size is required is that the pervasive caching contributes the most to compensating lossy wireless channels. While caching also improves the data availability for the subsequent Interests sent by other requesters, Interests from different requesters spread over longer time and hence the performance gain is not significant as the performance is more sensitive to mobility than to cache size under random replacement policy. Therefore, a small cache that can accommodate the aggregate data traffic on the path at any time is sufficient to achieve the majority of performance gain.

The above is confirmed when looking at the cache hits in Figure 4.8. The cache hits is flat once the cache size is large enough, showing that the cache utilization does not increase with the cache size. Moreover, the aggregate traffic results in Figure 4.9 show that the larger cache size only reduces the traffic slightly and thus it is sufficient to use a small cache at every node.

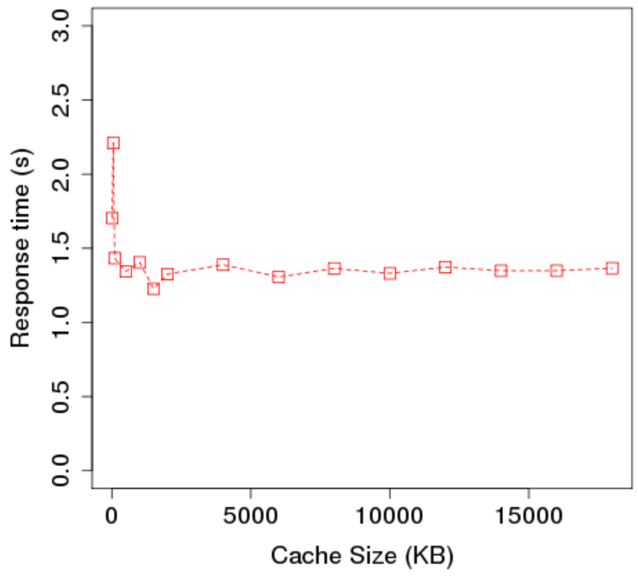

Figure 4.7: Cache size test: response time

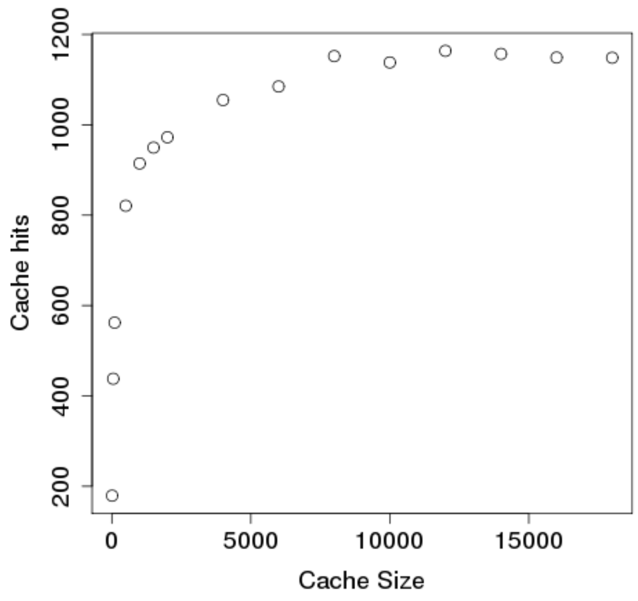

Figure 4.8: Cache size test: cache hits

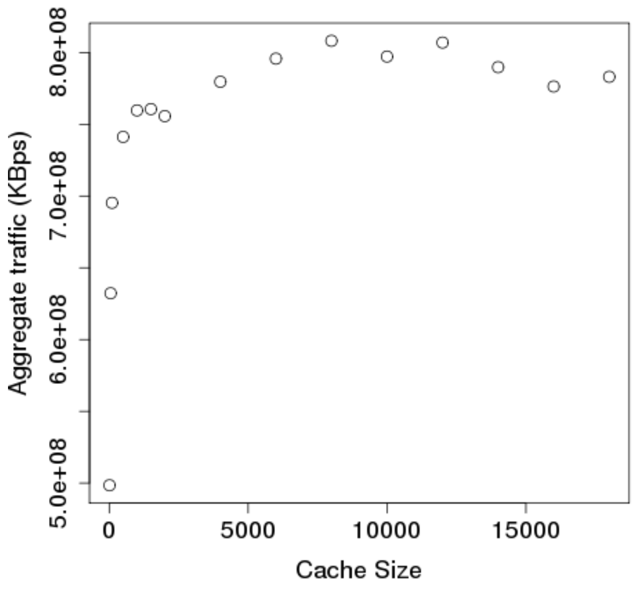

Figure 4.9: Cache size test: aggregate traffic

4.3.3 Performance gain with varying data object popularity

We further explore the performance with different data object popularity distribution. We vary the zipf distribution parameter from 0.2 to 1.0. As shown in Figure 4.10, the number of cache hits increase as the zipf parameter increases, that is, as users are more likely to request the same files. This is as expected since the more Interests are for the same files, the higher the cache utilization is. This also leads to higher completion ratio (Figure 4.11 and file retrieval ratio (Figure 4.12), since it is more likely for the users to obtain the popular data chunks from caches when the popularity distribution is more concentrated.

On the other hand, we observe an increasing trend of response time with zipf parameter larger than 0.8. This is due to the increasing congestion caused by cache storm due to more widely-spread popular files. However, the increment is slight and does not affect the overall performance significantly. We confirmed in Figure 4.14 that from the user's point of view, the goodput still poses an increasing

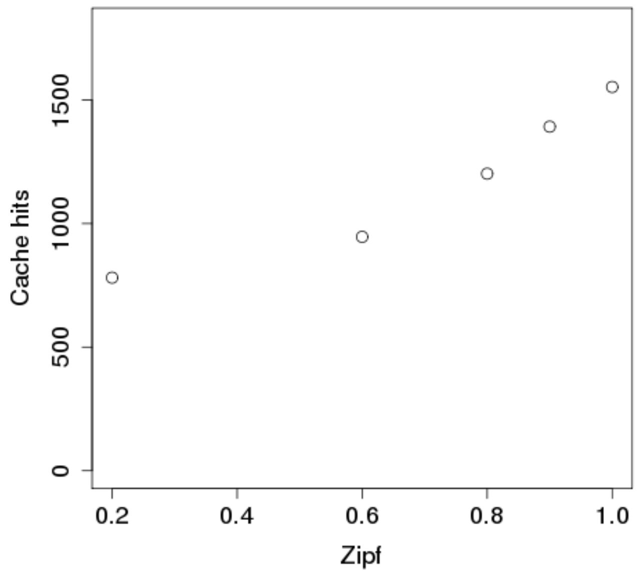

Figure 4.10: Zipf parameter test: cache hits

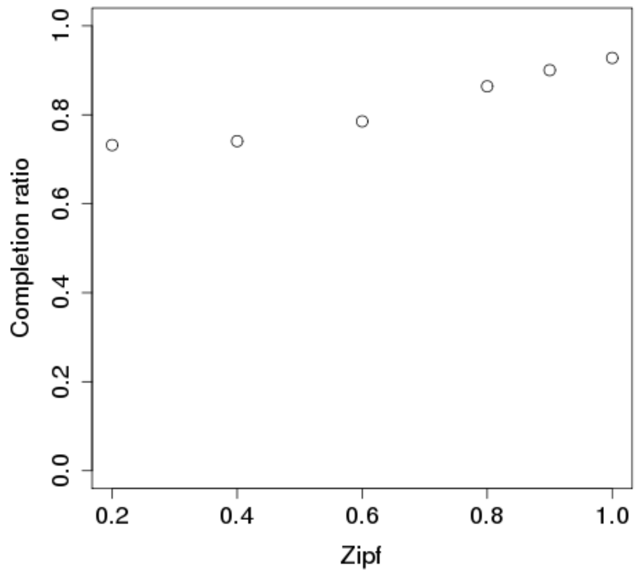

Figure 4.11: Zipf parameter test: completion ratio

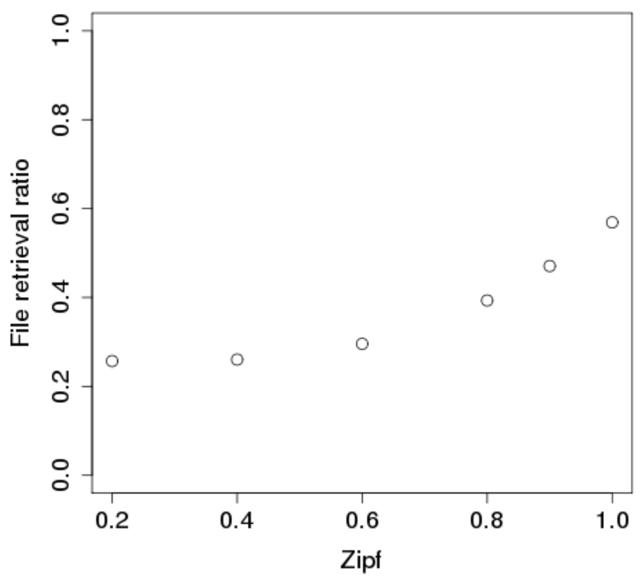

Figure 4.12: Zipf parameter test: file retrieval ratio

trend with the more concentrated file request distribution.

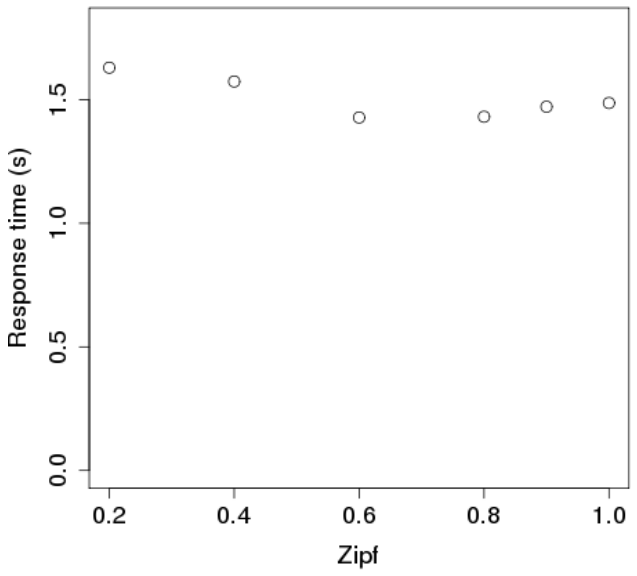

Figure 4.13: Zipf parameter test: response time

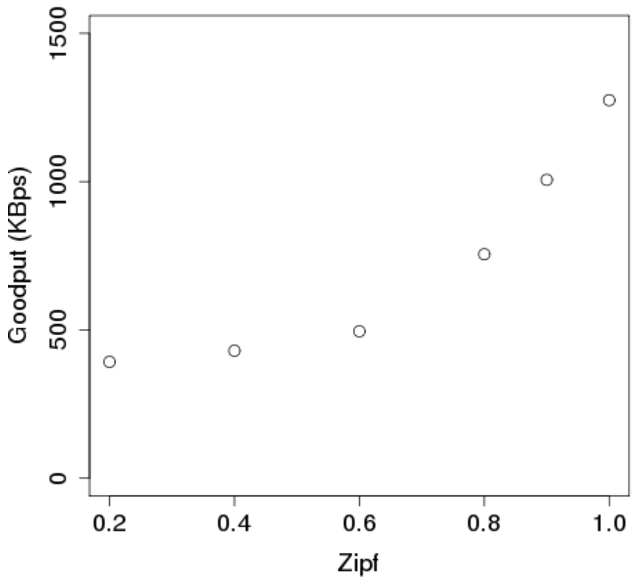

Figure 4.14: Zipf parameter test: goodput

CHAPTER 5

Context-Aware Content Discovery

5.1 ICN Content Discovery

Content discovery in ICN differs from IP network in the sense that there may be multiple locations of the same data object or chunk, and the number of data objects to locate is much larger than that in IP network, as in IP network the discovery design focuses on tracking only the hosts. Therefore, content discovery is an important playground for future ICN VANET research. The state-of-art ICN content discovery assumes data will be searched by flooding [19]. However, as the number of contents grows, this approach may lead to high overhead and congestion. In this chapter, we propose a context-aware content discovery to eliminate such overhead.

The ICN content discovery problem can be reduced to the routing problem of the first interest of a data object with unknown location. Therefore, it is in fact part of the name-based interest routing and relates most to the node discovery phase of the IP-based routing. We discuss below three content discovery paradigms for VANET inspired by reactive routing, proactive routing, and opportunistic forwarding.

5.1.1 Reactive content discovery

Reactive content discovery is accomplished via flooding. The first interest of a data object is flooded if the location of the data object is unknown. Once the data is found it is forwarded in reverse on the path(s) from which the interest arrived (breadcrumb routing), and the location of this data object will be recorded. The potential issue of reactive content discovery is that the frequent flooding and re-flooding to search new data sources may cause network congestion and nullify the caching benefits.

5.1.2 Proactive content discovery

Proactive content discovery is based on data object advertising. Flooding is not required since the object locations are pre-announced, and thus the flooding traffic may be eliminated and the searching delay is small. This way, the foreground search overhead is much lighter than that in reactive routing. On the negative side, there may be significant background advertising overhead. An ICN would require storing location information of at least 10^{12} data objects [57]; maintaining location information for all content names means significant cost in vehicular network. If names are hierarchical, as postulated in [?], one can perform prefix routing (as currently done in the Internet for DNS Domain Names and for IP addresses) rendering the problem more manageable. However, only a selected set of prefixes can be advertised.

5.1.3 Opportunistic content discovery

The two previous approaches require an active (i.e., reactive or proactive) content discovery phase. To avoid the associated overhead, an approach called opportunistic forwarding, or carry-and-forward scheme, has also been pursued. Requests and advertisements are not flooded or propagated proactive to the entire network. Rather, they are disseminated hop by hop through "opportunistic" node encounters. This scheme allows total control on propagation overhead. The opportunistic scheme significantly reduces the request flood and location announcement overhead. However, it may also introduces significant latency and was originally developed strictly for delay tolerant applications.

5.2 Context-Aware Content Discovery

Our simulation results in 4.3 show that while pervasive caching is critical in urban VANET, nearest-replica routing does not significantly improve the performance. This leads us to the conclusion that the overhead of maintaining locations of all content holders is not necessary. For proactive content discovery, it is reasonable to only keep track of the location of the data sources. Therefore, we suggest only the data sources advertise their application prefix proactively. This way, the benefits of proactive content discovery is maximized while the overhead is minimal. However, as the number of applications may still increase, it is not appropriate to assume all application prefixes can be advertised. Therefore, we propose a hybrid content discovery framework for VANET ICN applications to improve the scalability.

We expect the ICN applications will be categorized by their popularity. Popular data services are generally accessible by most users at least in a local geographic scope and the target services are considered location-based, and thus one can assume that such a service has a well-known service name prefix. Emergent announcements and map services are examples of this category. Depending on the number of applications available in the network, the application prefixes advertised are limited to the most popular k registered applications in an area. In this case, the volume of advertisement traffic can be controlled. We propose a proactive content discovery method, Hierarchical Bloom Filter Routing (HBFR), in section 5.3. The other applications must be accessed by reactive content discovery, that is, by flooding. However, the flooding range must be limited to prevent excessive bandwidth consumption. We propose a hybrid reactive and opportunistic content discovery method, Last-Encounter Content Routing (LER) in section 5.4, for the searching of less popular application data.

Note that although based on the results in 4.3, it is unnecessary to keep track of cached copies since nearest-replica routing performs similar to data origin-only routing in urban scenario. However, a service may be provided by multiple provider nodes. Therefore, the multi-source nature of ICN still needs to be considered in the content discovery design.

5.3 Proactive Content Discovery: Hierarchical Bloom-Filter Routing (HBFR)

5.3.1 Bloom Filters

We speculate that popular content will be offered by a few fixed or mobile providers and the popularity of these data services follows Zipf's distribution as the web traffic does [58][55]. With these premises, it is reasonable to use Bloom-filters [59] as a content digest to announce only the popular data service prefixes.

Bloom-filter (BF), which is widely used in applications such as Internet caching and P2P content discovery [59][60][61], is a probabilistic data structure that is used to test whether an element is a member of a set. A BF is an m-bit array. To add an element to a BF, the element is fed to k hash functions which map to k array positions and these k bits are set to 1. The existence of an element can be queried by examining the k positions of the array. BFs can be easily integrated/merged by a bitwise OR operation. False positives are possible for BF queries, while false negatives are not. Suppose the number of elements added is n, the false positive probability of a BF is given as

$$(1 - e^{-(\frac{kn}{m})})^k \tag{5.1}$$

In our case, the elements are the prefixes of data services. Due to the multi-path nature of ICN, a false positive may cause extra transmissions in data discovery, but this excessive overhead can be negligible as it is proportional to the false positive rate of the employed BF hash algorithms. Assuming $n = 100$ data services exist in the area, the false positive probability is about 0.86% with an 1Kbit BF array using 5 hash functions. Smaller overhead and better scalability can be achieved with more advanced BF designs [62][63]. In our implementation, we use the original BF algorithm proposed in [47] and 5 hash functions elf32, SDBM, DJB, DEK, and BP to demonstrate the lower bound of the potential performance.

5.3.2 HBFR Overview

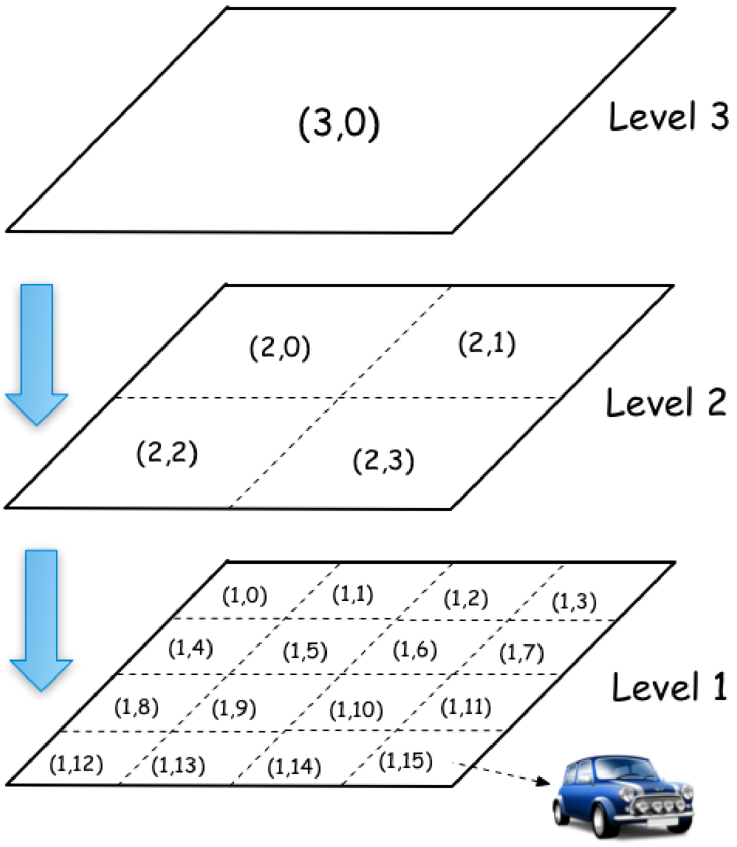

Figure 5.1: Simple partitioning example

In HBFR, the urban map is hierarchically organized into geographical partitions and the vehicles are likewise partitioned in corresponding clusters. All vehicles are equipped with GPS and thus they are time-synchronized and know which partition they are in. In Figure 5.1, a three-level rectangular partitioning that fits a Manhattan grid topology is shown. We denote the partition j of level i as $\{i, j\}$. The vehicle in Figure 5.1, according to its GPS coordinates, decides that its corresponding partitions are $\{1, 15\}$, $\{2, 3\}$, and $\{3, 0\}$. In practice, leaf partitions can be designed to correspond to road segments. Segments joined by road intersections form the next level partitions. Each vehicle reads the number of levels and the cluster maps when it enters the local VANET.

BFs are used to advertise the presence of private data services' prefix. Each node maintains a node BF summarizing its local content; the node BFs are then used to construct the partition BFs, which are the content digests of the corresponding areas. Each node stores (1) node BFs originated from all nodes in the same leaf partitions with it and (2) the partition BFs of its own partition and the sibling partitions at every level. In other words, node BFs are disseminated to all nodes in the same leaf partition, and the level i partition BFs are disseminated to all nodes in the same level i+1 partition. In Figure 5.1, partition BF$\{1, 15\}$ is disseminated to all nodes in $\{2, 3\}$.

The content search mimics a DNS query that is refined level by level. We illustrate the procedure by example. Since the vehicle shown in Figure 5.1 is in partition $\{1, 15\}$, it has the node BFs from nodes in partition $\{1, 15\}$, and partition BFs $BF\{3, 0\}$, $BF\{2, 0\}$, $BF\{2, 1\}$, $BF\{2, 2\}$,$BF\{2, 3\}$, $BF\{1, 10\}$, $BF\{1, 11\}$, $BF\{1, 14\}$, and $BF\{1, 15\}$. Suppose it receives an interest for content that happens to be in $\{1, 0\}$, the vehicle starts from examining node BFs it holds and then level 1 BFs. Since it does not find any match at the leaf level, it checks the higher level filters and finds a match in $BF\{2, 0\}$. It forwards the query to partition $\{2, 0\}$ using modified SSP. At this point, the interest carries the destination partition ID $\{2, 0\}$ and targets the anchor coordinates, which is defined as the center of the partition in our implementation, of partition $\{2, 0\}$. Each eligible forwarders, while receiving this interest, examines their lower-level BFs with the content prefix. The destination partition may be replaced by a lower level partition at any time if the eligible forwarders find a BF match in their database. In the worst case, once the interest is propagated to nodes in partition $\{2, 0\}$, the nodes find a match in $BF\{1, 0\}$ and then updates the interest destination to $\{1, 0\}$. The procedure repeats hereafter and the nodes in $\{1, 0\}$ finds a node BF that matches and redirect the request to the producer node. Data follows the breadcrumb of the request back to the data consumer and carries the producer location, which will be used as the destination of the following requests directly. Meanwhile, all relays in between cache the producer location and this information is used to assist other content search requests.

5.3.3 Advertisement Packet Generation

We assume the service providers can judge if the content they provide is popular. The popularity of services is decided based on prior knowledge and the statistics collected. If the provider determines that the application prefix is unpopular based on number and diversity of request it received, it stops advertising it in the BF.

All nodes periodically summarize their content to create their own node BFs. Only the data service providers should include the prefixes in their node BFs. At the time when a node generates its node BF, it also creates a local-view partition BF for each of its ancestor partitions by merging all BFs of this ancestor partition's children partitions/nodes. The node then may send out its node BF and partition BF advertisement if appropriate. In order to reduce the control traffic, a node only populates/forwards a BF if it is "fresher" than the partition BFs that have been circulated in the area.

Node Advertisement Packet
Node ID
Node Geo-Coordinate
Freshness Index
Node BF

Figure 5.2: Node advertisement packet format

Partition Advertisement Packet
(Level, Partition-ID)
Partition BF
Freshness Index

Figure 5.3: Partition advertisement packet format

The packet formats of BF advertisement packets are shown in Figure 5.2 and 5.3. The freshness of a BF is indicated by the freshness index carried in the packet. For node BFs, the freshness index is its creation time at the originator (in a loose granularity, say in second). The larger the freshness index is, the newer the information in the BF is. For partition BFs, the freshness index of $BF\{i,j\}$, $F_{i,j}$, is calculated by the following heuristic:

$$F_{i,j} = \max_{\forall k, P(\{i-1,k\})} \lceil \frac{F_{i-1,k}}{G_i} \rceil \tag{5.2}$$

where $P(x)$ represents the parent partition of partition x. Note that $F_{i-1,k}$ represents the freshness index of the lower level BF $BF\{i-1,k\}$. A ceiling is taken as we wish to define a looser granularity of the freshness index (time) for higher level. For example, while the node BFs may record a creation time in seconds, the partition BF may map its freshness index in the unit of 5 seconds at level 2 ($G_i = 5$). In this case, node BF freshness index 0, 1, 2, 3, 4 all map to the partition BF freshness index 0. Equation 5.2 states that the freshness index of a partition

BF is the creation time of its freshest children partition/node BF. In other words, the freshness of a partition BF is approximated by the freshness of the newest BF among all lower-level BF aggregated. Note that if a node receives advertisements of the same nodes from different paths, the staler or equivalently fresh advertisement, i.e. with smaller or equivalent freshness index, than the one recorded locally at the receiving node are discarded to reduce redundant processing and control traffic.

To balance the tradeoff between overhead and accuracy, the BF initiation frequency is set proportional to the degree of change of current BF advertisements. The degree of change is defined as the number of bits newly set/removed compared to the previously generated BFs. All BFs have a pre-defined lifetime, which is two times of their announcement frequencies. Expired BFs are excluded from both the partition BF aggregation and the content search.

Note that different from the previous approach [64], there is no cluster head in each partition to supervise the BF aggregation. The aggregation is entirely distributed. The major advantage of the distributed BF aggregation is robustness to mobility. In a highly mobile network the cluster head may be short lived and must be reelected when it drifts out of its region. Moreover, in an intermittent VANET, the cluster head may become isolated from other nodes thus jeopardizing the proper transmission of BFs to upper layers. HBFR is robust to node isolation and intermittent connectivity, since it percolates content information via BFs in parallel among all nodes.

5.3.4 BF Propagation: local-view vs. global-view BFs

The BFs must be periodically refreshed to account for content withdrawals due to mobility across cluster boundaries. In order to keep the BFs at each node synchronized with low control traffic overhead, while a node must maintain a local-view partition BF by aggregating lower-level BFs it currently has, it also keeps a global-view version of partition BFs. The global-view partition BF represents the freshest partition BF that is being circulated, and may differ from the local-view partition BF.

While node BFs are disseminated to all nodes in the same leaf partition of the BF originator by simple restricted flooding without further processing, the global-view partition BFs are maintained in a distributed manner. As all nodes create their own local-view partition BFs for its corresponding partitions, nodes only propagate global-view partition BFs. When a local-view partition BF is created, the node compares the freshness index of the local-view partition BF with that of the current global-view partition BF. If the local-view partition BF is fresher, it replaces the global-view partition BF and broadcasts the advertisement; otherwise, the local-view partition BF is kept on file but not propagated. When receiving new partition BF advertisements from other nodes, the same principle holds. A node only forwards a received global-view partition BF if both of the following conditions are satisfied:

1. The BF is a content digest of its own partition or sibling partitions.
2. The received copy has a larger freshness index than that of its local global-view partition BF copy of the particular partition.

Note that the nodes also receive and forward the global-view BFs of the sibling partitions (following the same principle of freshness), but do not maintain local-view BFs of them. In this way, the global-view partition BFs are maintained in a distributed but synchronized manner, and the advertisement overhead is significantly reduced.

During the content searching phase, both local-view and global-view partition BFs are used to compensate the potential information loss of the freshness heuristic in case of packet losses. In addition, nodes utilize all valid BFs they have received to best utilize the caching ability. Since all packets are broadcast, if a node overhears the advertisement of a remote partition, the advertisements are kept in its storage and may be used for local content search.

5.3.5 Modified SSP Forwarding

In the original SSP described in 3.5.3, the rebroadcast is prioritized purely based on the eligible forwarder's distance to destination. In HBFR, nodes that have more updated destination information (i.e. the nodes found a match in BF of level lower than that of the current destination recorded in the packet) must be given chances to transmit a packet. Therefore, we define nodes satisfying one of the following conditions as eligible forwarders:

- Relays who have lower level destination information.
- Relays who are closer to the destination carried in the packet than the last hop is.

Once the eligibility of forwarding is decided, an eligible forwarder sets its rebroadcast timer based on the prioritization function. The other vehicles are ineligible and automatically drop the received packets. Nodes that have a destination location update are given highest priority and will set their expiration timer randomly within $[0, T_{update}]$. The remaining eligible forwarders are prioritized by their normalized distance to the destination; thus, their rebroadcast timer is calculated as follows:

$$T = T_{update} + (T_{dist} - T_{update})\frac{d_{ref}}{d_{max}} \tag{5.3}$$

5.3.6 Scalability Analysis

The HBFR approach includes three sources of potentially non scalable overhead: BF size storage overhead; BF dissemination overhead and query search overhead.

We contain the BF size storage overhead by limiting the entries to popular content prefixes. With the hierarchical naming design, the number of advertised content prefixes is limited in a geographical area. Note that many data service originators may share the same prefixes (e.g. Internet gateway service), and with the popularity constraint the number of advertised prefixes does not grow with the density of nodes.

The BF dissemination overhead is well-controlled by the freshness index heuristic. Although fresher global-view partition BFs may be created from time to time, the definition of freshness index limits the number of global-view partition BFs being populated. This, of course, comes with the price of large query search overhead, as the global-view partition BFs may not contain all information if the originator missed some of the lower level BFs. However, this issue is compensated by the fact that all nodes in the same partition keep their local-view partition BFs. In this case, as long as there are some nodes that hold an accurate local-view partition BFs exist near the anchor coordinates of the upper level partition, the packet can be redirected to the correct lower level partitions.

Query search (interest routing) overhead is proportional to $\sqrt{N}$ if the content is uniformly distributed over the urban grid. In most practical cases in VANET, the content is location relevant (e.g, traffic jam information) and thus the query is satisfied in the local partition, reducing request routing overhead to $\log(N)$, where N is the name space size. HBFR offers the advantage of aggregation, with the drawback of false positives. However, the query search overhead due to the false positives is negligible with proper popularity constraint.

5.3.7 Simulation

5.3.7.1 Simulation Settings

We conduct a simulation using Qualnet 6.1. The simulation time is 600 seconds. All experiments are repeated 20 times for all scenarios using different random seeds.

We use a $2km^2$ Washington DC map from Tiger/Line file [50]. The vehicle movement traces in our simulation consist of 50 vehicles in a $2km^2$ area generated by SUMO [51]. For a comprehensive evaluation, we present the results of scenarios generated by three different random seeds (referred as scenario 1, 2, and 3 later on) with the same setting. To understand the effect of data caching, we consider situations where requesters retrieve the same and different data objects. We test two different types of popular data services: popular public data service and popular private data service. We assume popular private data service provides customized data for each requester, while the popular public data service provides data objects of all requesters' interests. The popular public data service scenario includes ten randomly assigned data consumers downloading the same 900MB file from one mobile data producer. The request sending rate is 1 per second for each consumer. All downloads start at the same time at 7 second after the BFs are stabilized. The popular private data service settings are the same as that of the popular public data services except ten data consumers request ten different files representing encrypted private data from the data producer.

We use IEEE 802.11a PHY and MAC, with data rates 24Mbps (transmission range 200 meters) and 6Mbps (transmission range 400 meters). Intermediate nodes are allowed to retransmit forwarded requests for up to two times. Data consumers may reinitate the request for up to four times. The retransmission interval is 100ms and 500ms for intermediate nodes and data consumers, respectively. For HBFR, the Bloomfilter size used in the simulation are 72 bytes. A two-level square partitioning similar to that in Figure 5.1 is used. The leaf partition size is $1000m^2$.

We compare HBFR with a baseline SSP which assumes zero content discovery cost and a reactive content discovery realized by SF. In addition to compare different content discovery mechanisms, we present the results for each routing mechanism with and without caching (denoted by noC) to analyze the effects of caching.

We measure (1) the response time, which is defined as the interval from each request sent to its corresponding data chunk received, and (2) the completion rate, which is defined as the ratio of the number of corresponding data received at the receiver side over the number of requests sent. We also measure request, data, and control traffic rates of the network, which includes the request, data, and control traffic forwarded by all nodes. Confidence intervals (CI) are reported.

5.3.7.2 Evaluation

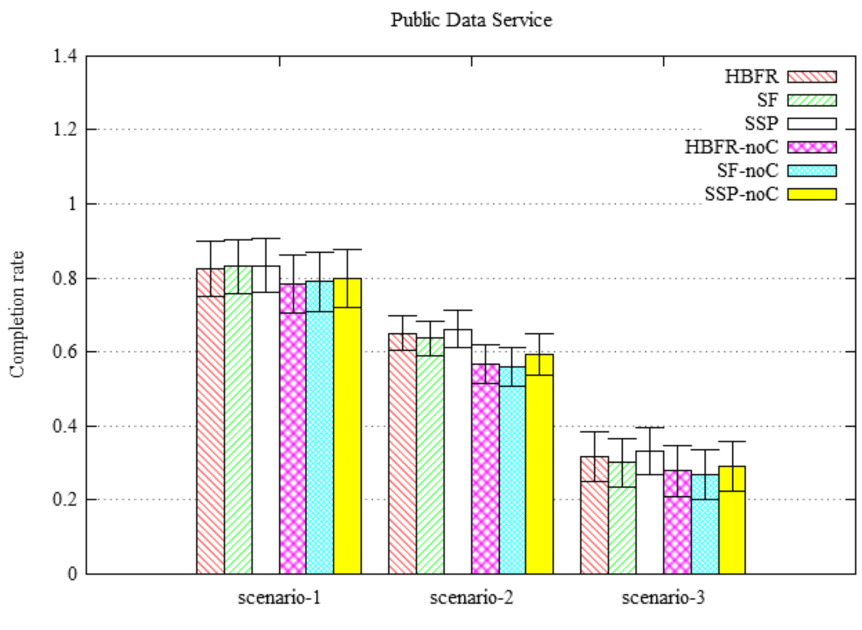

Figure 5.4: Completion rate: public data services (transmission range 200 meters)

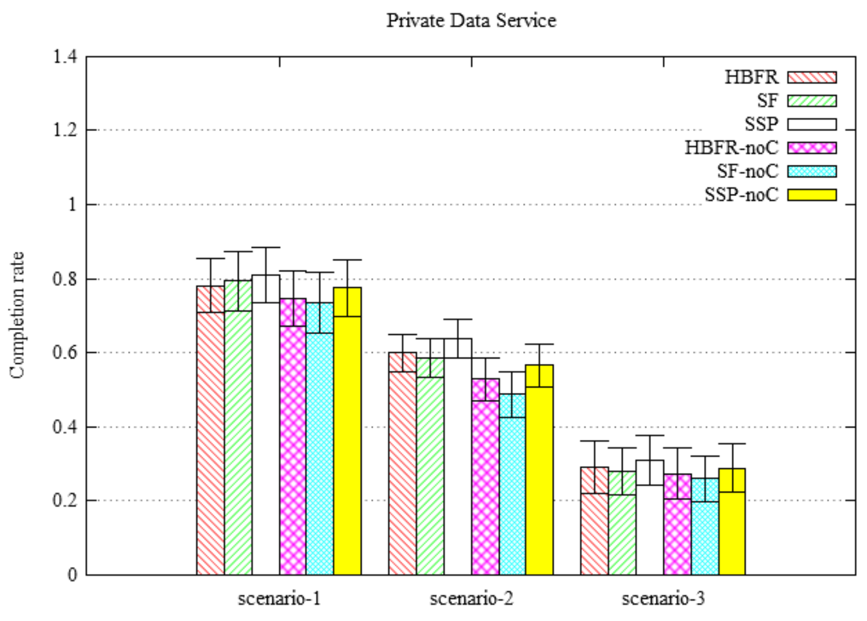

Figure 5.5: Completion rate: private data services (transmission range 200 meters)

The completion rates vary significantly with the data rate used. When the transmission range is 400 meters, the network is always connected. We observe nearly 100% completion rates for all approaches, with or without caching. Due to the limited space, the completion rate of 400m range is omitted. Figure 5.4 and Figure 5.5 show the completion rates results of popular public and popular private data services for transmission range 200 meters, respectively. With 200 meters transmission range, the network is intermittent and thus we observe lower completion rates for all schemes. We find SSP always achieves the highest completion rates as it has perfect knowledge of content locations at all times. Interestingly, even with the advertisement overhead and potential inaccuracy, HBFR still achieves higher completion rates compared to SF. The reason is that while overhead has been largely reduced, SF experiences more collisions due to hidden terminals created by multi-path forwarding in the all-broadcast system. The situation gets worse when caching is disabled. The completion rates are similar among different type of services with the help of intermediate node retransmission.

Therefore, we next investigate the performance in terms of overhead and latency.

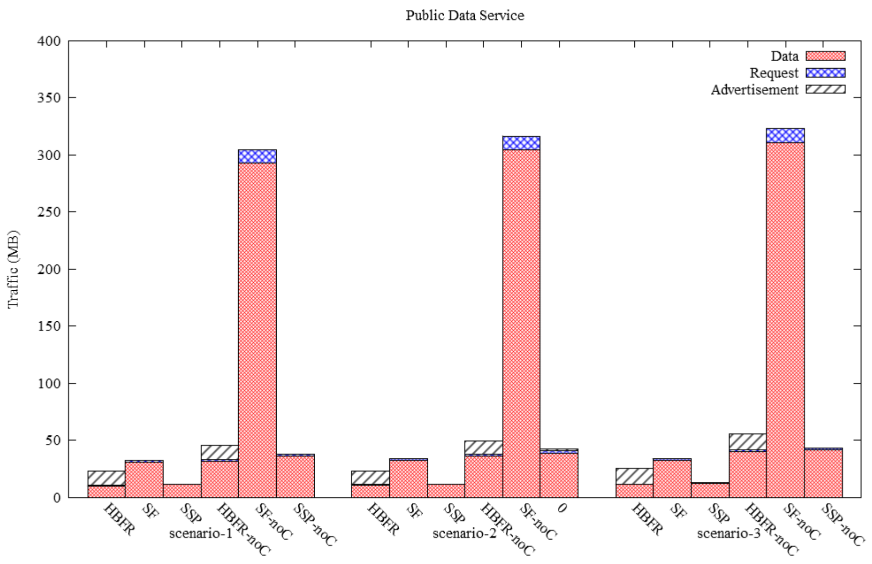

Figure 5.6: Traffic: public data services (transmission range 400 meters)

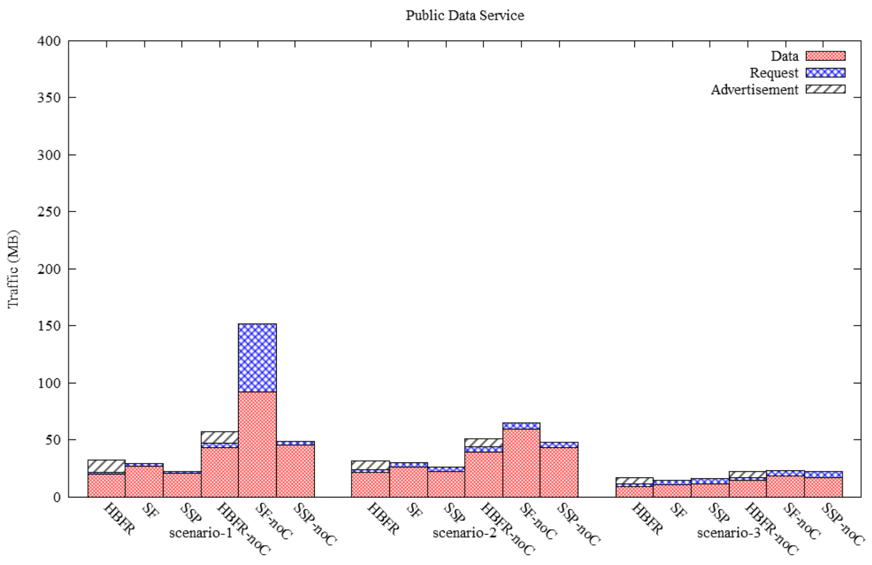

Figure 5.7: Traffic: public data services (transmission range 200 meters)

Figure 5.6 and 5.7 presents the composite traffic of popular public data services. Despite the request routing approaches, caching significantly reduces the traffic load for public data. Comparing HBFR and SF, when the network is always connected with 400 meters transmission range (Figure 5.6), HBFR creates less traffic even considering the BF advertisement overhead, and the volume of data traffic is reduced by more than 60%. The volume of data and request traffic transmitted by HBFR and SSP are comparable, showing the HBFR content discovery is nearly as efficient as the oracle content discovery of SSP. Note that the HBFR content advertisement overhead is constant in a stable network. Hence, if the popularity increases, i.e., the number of consumers increase, we expect the traffic saving of HBFR over SF will be more significant in a connected network. In Figure 5.7, we observe that SF induces less traffic in intermittent network than HBFR does as both the request and data tend to traverse multiple paths with HBFR under intermittency.

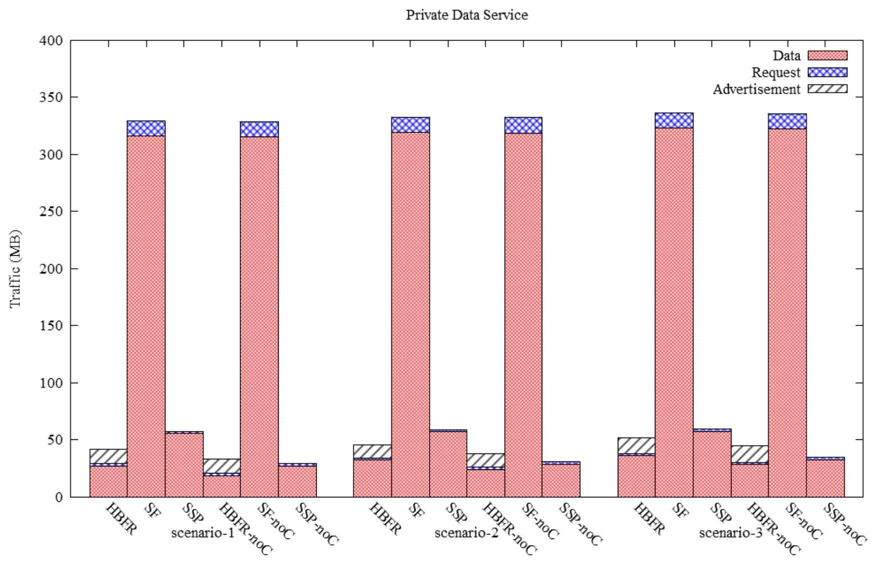

Figure 5.8: Traffic: private data services (transmission range 400 meters)

Figure 5.8 and 5.9 present the composite traffic of popular private data services. For both popular public data service (Figure 5.8) and popular private data service (Figure 5.9), SF induces very high volume of traffic. HBFR reduces the traffic by up to 85% in connected scenarios.

Figure 5.10 and 5.11 show the response time of public data services. Interestingly, in Figure 5.10, we observe that the response time of SF with caching is the

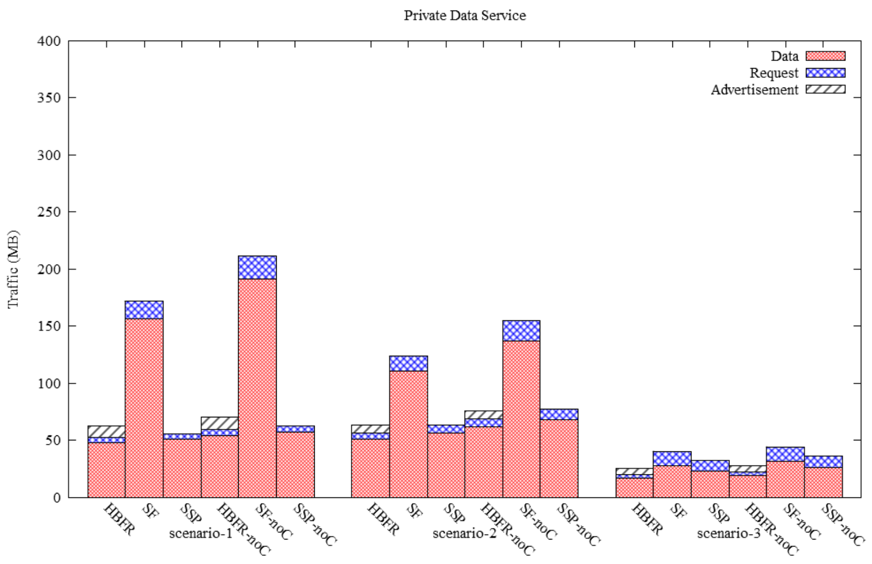

Figure 5.9: Traffic: private data services (transmission range 200 meters)

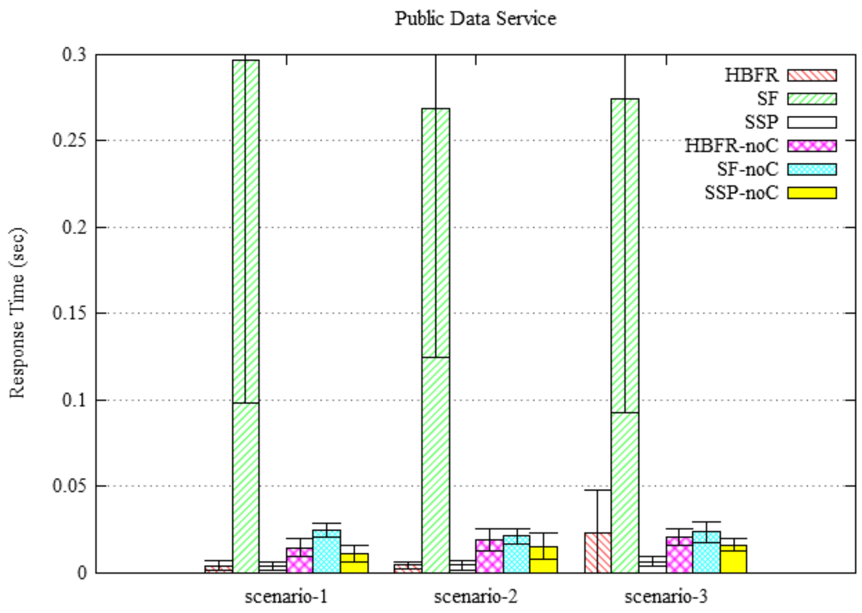

Figure 5.10: Response time: public data services (transmission range 400 meters)

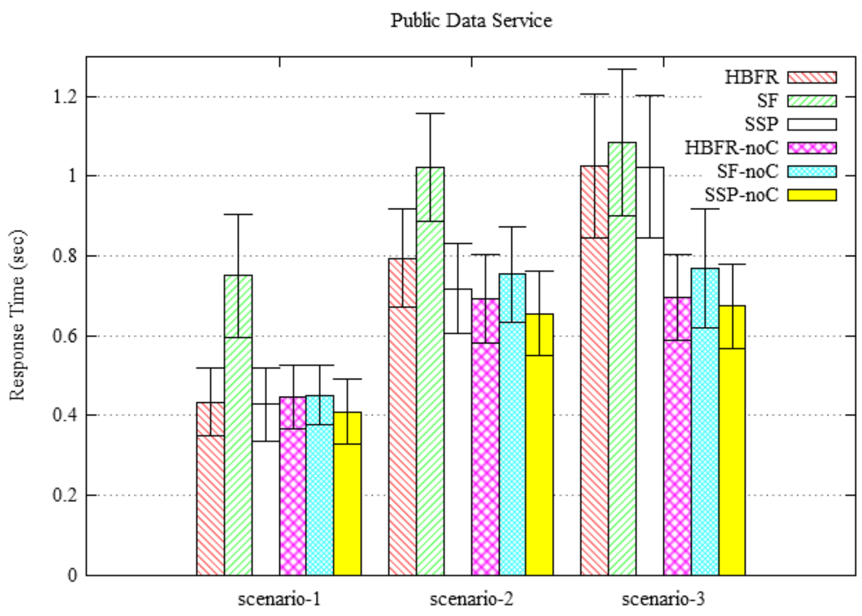

Figure 5.11: Response time: public data services (transmission range 200 meters)

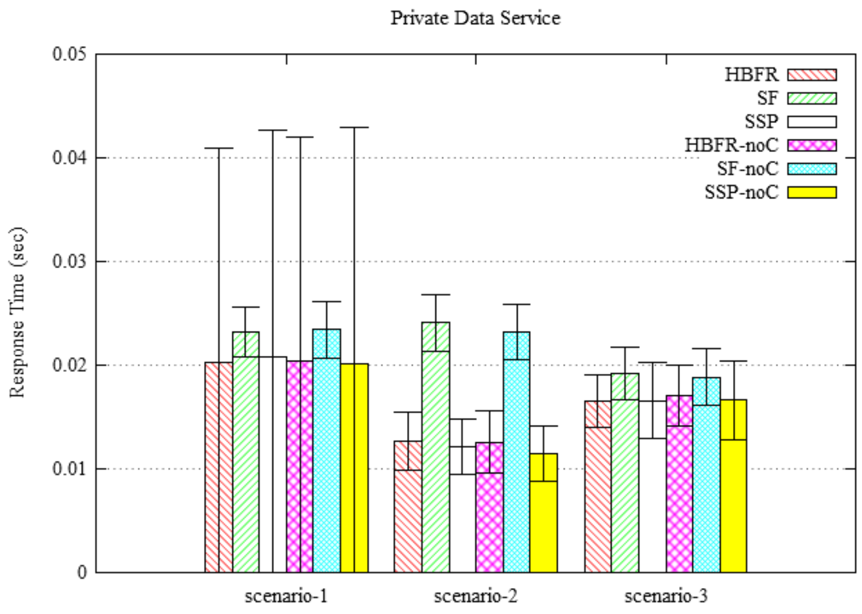

Figure 5.12: Response time: private data services (transmission range 400 meters)

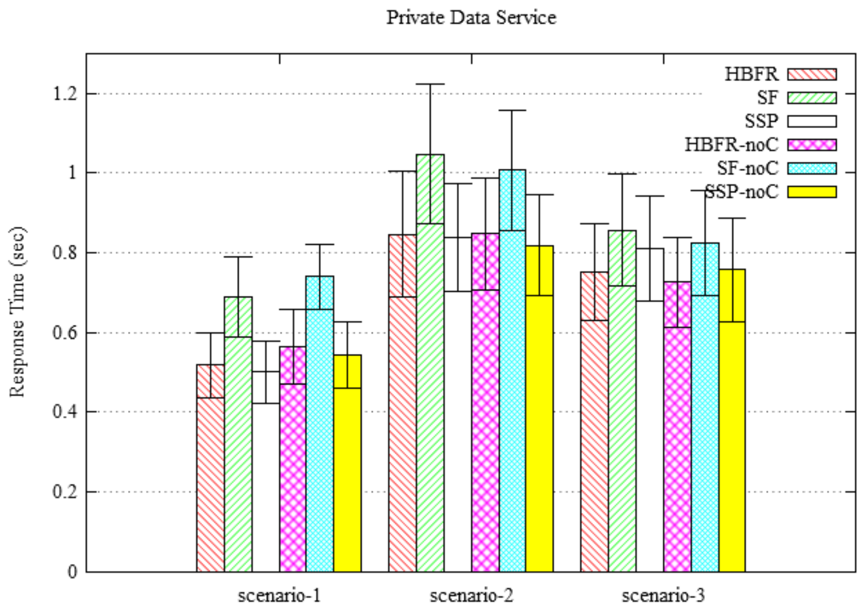

Figure 5.13: Response time: private data services (transmission range 200 meters)

largest due to the contention of multiple caches and hidden terminals. Several retries are required for SF consumers to receive data. In case that the transmission range is limited to 200 meters (Figure 5.11), the improvement of HBFR over SF is reduced. In both cases, the response time of HBFR and SSP are comparably low. Note that HBFR traverses longer paths than SSP does since HBFR requests may be sent via the geographical hierarchy, and hence we expect HBFR has slightly higher response time. Figure 5.12 and 5.13 shows that HBFR remains advantageous compared to SF and achieves comparable performance as SSP does for popular private data services.

5.4 Reactive Content Discovery: Scalable Opportunistic VANET Content Routing With Encounter Information

5.4.1 Protocol Design

For the applications not popular enough to be maintained using HBFR, we propose Last Encounter Content Routing (LER). LER is a hybrid reactive and opportunistic approach. We enhance the flooding-based reactive interest routing by integrating the last encounter information concept [65][66], which helps vehicles gather the location of content providers. In LER, the interest flooding is restricted to k hops to prevent excessive overhead. In the mean time, LER opportunistically keeps track of content locations using last encounter information obtained by one-hop beacons. Each node maintains a database of last-known content locations. The locations are updated only when encountering the data source. Note that the information is not guaranteed to be accurate, but it is likely the interest will reach the content holder.

LER has two phases. In the first phase, the location of the content is unknown. Thus, the naive requester floods the Interest to search for the content location. The second phase starts once the Interest reaches a relay that has the location information of the particular content name requested. At this point, the routing module stamps the destination location in the Interest and afterwards the Interest routing switches from flooding to geo-routing. During the geo-routing process, the relay nodes keep updating the destination location if they have newer information.

We use the following example to further explain the idea. Suppose a vehicle A wants to retrieve a file f from an application *app*. While A does not know who holds the content, vehicle B is the data source of application *app*. Each vehicle maintains two data structures: content list and Last Encounter List (LEL). The content list, which is periodically advertised to one-hop neighbors, summarizes all the contents the node itself has. When vehicles receive their neighbors' content lists, they merge the information carried within the lists into their LELs. Each entry of the LEL includes the content name, the provider ID, the encounter location, and the encounter time. Note that (as a difference from [65]) one content service can be provided by multiple vehicles, so there can be multiple entries for the same content name, each of which represents a different vehicle.

Application Prefix	**Vehicle ID**	**Location**	**Time**
app	B	(x,y)	12:00pm

Table 5.1: The format of Last Encounter List (LEL)

Suppose B previously broadcasted its content list to its nearby vehicle C at 12:00 pm at location (x, y). The LEL of C is shown in Table 5.1. That is, vehicle C can provide an approximate destination location for application *app*. Suppose now vehicle A broadcasts its Interest *app/f*. The Interest is received by all nearby vehicles, C and D. C and D then search their LELs for *app/f* using prefix matching. As vehicle D cannot find any entry matching this name, it prepares to broadcast this Interest to its neighbors to continue the search. However, C has found the match. It adds the destination location geo-coordinates, destination nodeID, and encounter time to the Interest packet, and send the Interest packet out by opportunistic geo-routing, as discussed in the next section. Upon receiving Cs Interest, D realizes the Interest is delivered by geo-routing and carries content provider information. It thus aborts the rebroadcast.

If there are multiple providers for one content, there can be many strategies to choose the vehicle to serve the Interest. In our implementation, we randomly choose one vehicle to serve the Interest for the purpose of diversity and load balance.

5.4.2 LER timer-based forwarding

Our LER also uses the timer-based forwarding. As in HBFR, nodes that have more recent location information than the one carried in the packet set their expiration timer randomly within $[0, T_{update}]$. Since the highest priority is always given to forwarders who have relatively recent location information, the random timer for other nodes is set to be larger than T_{update}.

In phase 1 (the flooding-based content search phase), the interest propagation relies on a modified SF algorithm in which the waiting time of a node is

$$T_f = T_{update} + (T_{dist} - T_{update} \frac{(D_{max} - min(D_{max}, D_{transmitter}))}{D_{max}} \tag{5.4}$$

Phase 2 (the geo-routing phase) is triggered when a provider's geographic location is found. The geo-routing is done by modified SSP, in which the waiting time is set by equation 5.3. In addition, the rebroadcast is only cancelled if a node does not have newer location update for the interest.

5.4.3 Simulation

5.4.3.1 Parameter exploration

We first study the relationship between LEL exchange frequency and the efficiency of content search. The simulation parameters are as follows:

- MAC and PHY parameters: We use the 802.11 Adhoc WIFI module defined in ns-3. The wireless transmission range is set to be 150 meters. The maximum expiration time T_{dist} is 0.1 second. We use CORNER propagation model [52] to simulate signal loss in urban environments.

- Urban environment and mobility model: The urban map we uses comes from TIGER/Line files [50] collected by Census Bureau. We generated a 120s mobility trace using SUMO [51] to simulate practical vehicle movements.

- Application: For each trial, we randomly choose one vehicle as content consumer and one vehicle as content provider. The producer publishes its service name prefix */prefix* to its neighbors using LEL application. The consumer requests contents named */prefix/seq* every 1 second, in which *seq* starts from 0 and increases by 1 each time. That is, the consumer sends Interests named */prefix/1*, */prefix/2*, , etc. All requests are served by the provider based on longest prefix matching.

We set LEL exchange period T to be 1s, 5s and 10s, and examine (1) the number of flooding hops before switching to geo-routing and (2) the total number of hops an Interest traverses.

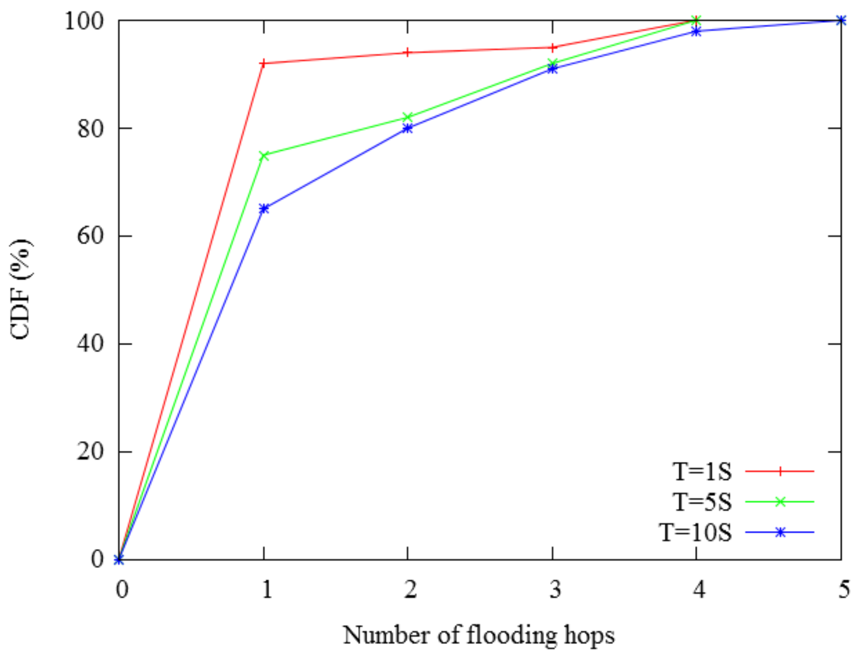

Figure 5.14: LEL exchange frequency vs. hops in phase 1

In Figure 5.14, we can clearly see that when LEL exchange frequency increases, the number of Interest flooding hops during search phase decreases significantly: when $T = 10s$, 63% of Interests can find the provider within just one hop. When $T = 1s$, the probability of locating provider in one hop increases to 92%. This is because the more frequent vehicles exchange LEL with each other, the more information each vehicle collects, and therefore the more likely a provider can be located. Note that the results above are normalized over the number of interests reached the data holder.

5.4.3.2 Performance comparison

We then evaluate the performance of LER by comparing it with SF, SSP, and GHT. GHT [67] is a well-known content storage method designed for mesh networks. The main idea of GHT is to hash a certain key k into geographic coordinates. In our implementation, the key k refers to a unique application prefix. Therefore, the prefix is hased to a geo-coordinates (x,y). Only those nodes whose distance to (x,y) is within d meters have to keep the location of the provider, where d is a small range ($d = 100$ in our experiment). The content provider advertises its application prefix list and location to the hashed geo-coordinates every T seconds.

This set of experiment is conducted using Qualnet 6.1. We use a $2km^2$ Washington DC map from Tiger/Line file [50]. The vehicle movement traces in our simulation consist of 50 vehicles in a $2km^2$ area generated by SUMO [51]. The data rate is 6Mbps. We simulate 10 randomly selected data consumers downloading 900MB files from 10 different applications on one provider node. T is 1 second. The simulation continues for 600 seconds and all experiments are repeated 10 times with different seeds.

The completion ratio results are shown in Figure 5.15. LER, SF, and SSP

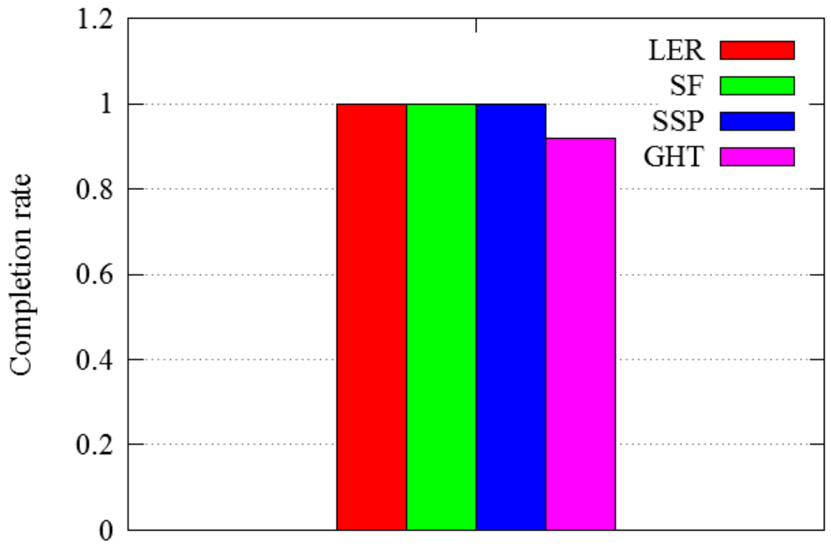

Figure 5.15: LER simulation results: completion ratio

all achieve 100% completion ratio in this experiment as an end-to-end path can be easily found. GHT has a completion ratio 8% lower, since nodes around the hashed locations may move away. In such case, the content providers cannot be located and the Interest will be dropped.

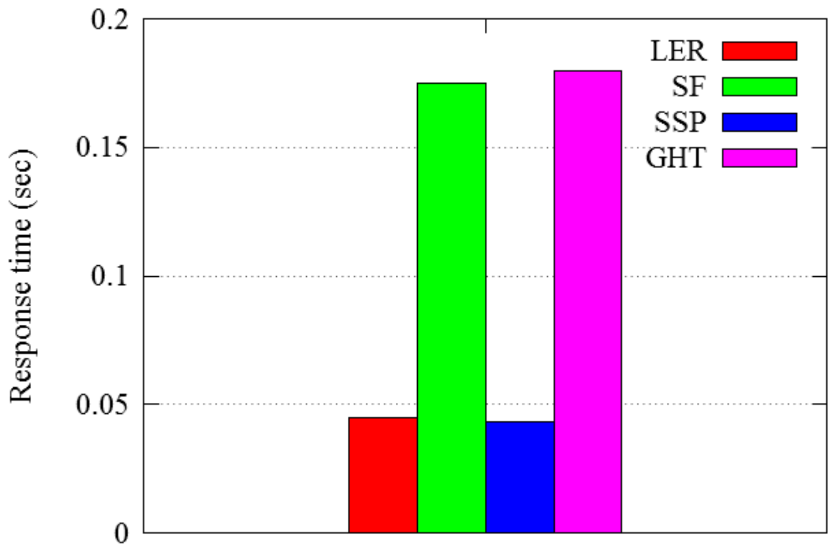

Figure 5.16: LER simulation results: response time

The response time results are presented in Figure 5.16. As expected, LER reduces the response time of SF's as it traverses a single path instead of flooding the Interests. On the other hand, the response time is only slightly higher than SSP's, meaning it only traverses a few hops more than SSP does. We also observed GHT has longer response time due to the effects of mobility. Interests must be retransmitted when the packets are dropped due to unavailable location servers.

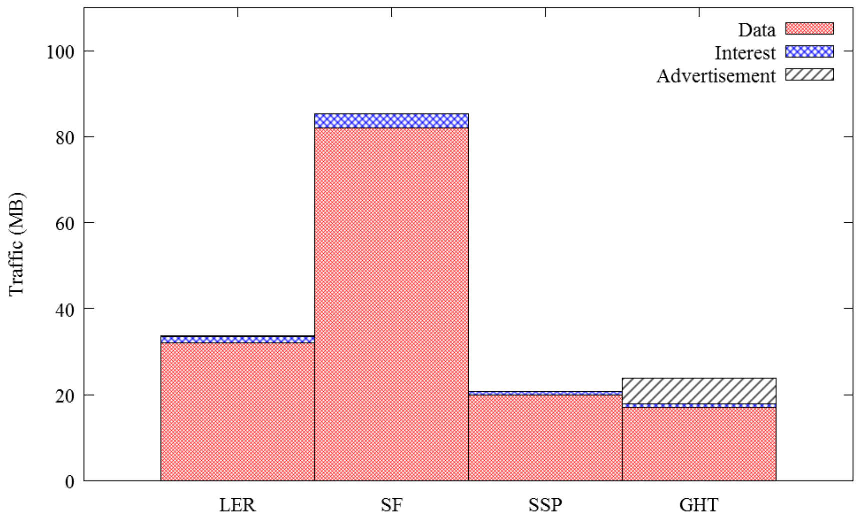

Figure 5.17: LER simulation results: composite traffic

Finally, the composite traffic is shown in Figure 5.17. We observe that LER reduces the SF traffic by about 60%. In addition, its advertisement overhead is minimum for this scenario. GHT's traffic is higher than SSP as it does not traverse the shortest path. However, LER achieves a good balance between the high completion ratio of flooding-based reactive content discovery and small control overhead and short response time of proactive content discovery.

CHAPTER 6

Disruption-Tolerant ICAN

In this chapter, we first describe in more details the DT-ICAN design for sparse VANET ICN. Later, we study the applicability of network coding in DT-ICAN to improve the file retrieval efficiency in extremely sparse networks.

6.1 DT-ICAN In Depth

6.1.1 Data retrieval in DT-ICAN

In DT-ICAN, data propagation is triggered by DTN-request instead of per-chunk pull-interest to prevent excessive usage of bandwidth. There are two major procedures in this process to control the bandwidth consumption: request generation, data object prioritization, and randomized chunk transmission. The receiving node uses request generation to control the amount of data transmission triggered; the sending node uses data prioritization to decide the order of data object transmission in the limited contact duration and randomized chunk transmission to reduce redundant transmissions from different forwarders.

6.1.1.1 Request generation

Since node-interest and DTN-request are both represented by Bloomfilters, the most efficient way to generate a DTN-request is by merging a node's own node-interest and the ones received from neighbors. This leads to a node priority-based request generation policy. Namely, the data to include in the request is decided by a node ranking locally computed. The node ranking policy can be flexibly defined. In our implementation, we assume all nodes have the same ranking for simplicity.

6.1.1.2 Data prioritization

When a node receives a request, it finds the requested data objects/chunks available in its content store. This procedure is called *request matching*. A list of available data chunks are compiled, and the node may order the data objects according to node priority and content attributes. Note that this is where the application/content awareness can help improve performance. In our simulation, data objects are prioritized in First-Come, First-Serve order.

6.1.1.3 Randomized chunk transmission

When a data transmission is triggered by the request, the available data chunks are sent to the receiving node in random order. The reason of the randomization is to improve the cached chunk diversity of the opportunistic network in case of short encounter durations. An alternative approach is to utilize network coding, which can further improve the performance under high intermittency and will be explored later. A handshake procedure is associated with each data chunk transmission to eliminate redundant transmission in the broadcast network. For each data chunk, the sending node first sends a Request-To-Send-Block (RTSB) carrying the chunk name to the target node. Upon receiving an RTSB, the target node sends a RTSB-Reply, which may accept or reject the block. If the block is rejected, a reject code is carried to indicate one of the three reasons: (1) The chunk is already received, (2) The complete object is already received, and (3) The chunk is being sent by other neighbors. The data is only transmitted if accepted. Once the target node receives the data, it acknowledges by an ACK. Note that all neighbors of the target node also updates their stored cache summaries based on the broadcast RTSB-Reply and ACK.

6.1.1.4 Utilizing partial multi-hop path

In case of partial multi-hop connectivity exists, we optimize the multi-hop data forwarding as follows. When a data block is received, a relay propagates the data back towards its original requester(s) by checking pending requests recently received from neighbors. If matches are found, the relay initiates a data transmission for the particular data. In this way, the data is delivered back to the original requestors via the trail of breadcrumbs. To eliminate redundant transmissions, if the data matches multiple interests, only one broadcast data transmission is initiated. This approach achieves the same benefit of per-content interest aggregation as in [7].

6.1.2 Node-Interest propagation

While DTN requests are only transmitted within one hop, the node-interests must be propagated so that the relays may start requesting data. Note that node-interests are not instantly flooded. Instead, they are broadcast periodically. This may affect system scalability when the number of reachable nodes is large. One potential solution is the following: The nodes periodically broadcast a subset of received node-interests using a given amount of capacity. The rest of the capacity is reserved for data requesting and transfer. The order of node-interest broadcasts are decided using the same node ranking algorithm as that in request generation. Note that while this greedy approach may lead to better scalability, we evaluate the performance using the simple periodic node-interest dissemination as a performance benchmark of the system.

6.1.3 Bandwidth reservation

In DTN mode, the packets are propagated in an epidemic manner. The epidemic routing consumes much bandwidth and may cause congestion when the network is partitioned but the isolated "islands" are dense itself. Therefore, it is important to restrict the bandwidth consumed by DTN packets. Therefore, we reserve $B_{DTNMIN}\%$ of the total bandwidth for DTN traffic and restrict the DTN communications to use no more than $B_{DTN}\%$ of the bandwidth when the aggregate traffic of all four forwarding engines exceed the available capacity.

6.1.4 DT-ICAN evaluation

We prove the necessity of the DTN mode, DT-ICAN, by comparing the performance of DT-ICAN and non-DTN ICAN, which is referred as ICN in the following, in sparse scenarios. For fair comparison, the ICN interests are flooded using SF to guarantee all reachable nodes will receive the instantly propagated interests; relay nodes are allowed to retransmit a packet by up to 4 times; the relay retransmission interval is 200ms; the requester continuously re-initiate its interest if data is not received every 1 second. In addition, each ICN data chunk is requested as soon as its previous chunk for the data object is received on the requester side to shorten delay. DT-ICAN broadcasts periodic control messages, i.e., cache summary, request, node-interest, every 2 seconds. All other settings for ICN and DT-ICAN are identical. We use IEEE 802.11a MAC/PHY. The data rate is fixed to 36Mbps. The transmission range is approximately 200 meters.

We measure three metrics: average file retrieval rate, average file completion delay, and average per-node traffic. The file retrieval rate is defined as the ratio of number of complete file received to total number of file requested. The file completion delay is the time elapsed since a file is requested/subscribed on the requester side until the file is received completely by the requester. To be precise, for DT-ICAN, the file is requested at the time the data object is added to the node interest of the requester; for ICN, the file is requested at the time the first chunk of the file is requested. The average per-node traffic, which represents the bandwidth consumption, is the average number of bytes each node transmits per second.

6.1.4.1 Synthetic trace with Washington DC map

We first simulate a synthetic trace for better understanding of the performance under various node densities. We downloaded a 2000m by 2000m map of the Washington, DC area made available by the US Census Bureau's TIGER database, and simulating mobility on the map using the Intelligent Driver Model with Intersection Management by VanetMobisim [68]. This model is complete with intersections and stop light rules to simulate realistic vehicular traffic. We generate three scenarios by varying the number of nodes between 50 and 100. The vehicle speed ranges from 11mph to 40mph. The stay time ranges from 5 to 30 seconds. In all scenarios, we randomly selected 9 data source nodes, each publishes a 128KB file. One randomly selected data consumer starts to download all files at 100 second. The simulation time is 1000 seconds. All experiments are repeated 10 times with different random seeds and confidence intervals are reported.

Figure 6.1 shows the file retrieval rate of the three scenarios. DT-ICAN outperforms ICN as end-to-end paths are not available all-time for all data sources. DT-ICAN's file retrieval rate increases as the number of nodes increases, and is able to retrieve all files in the 100 nodes case. On the other hand, ICN is only able to retrieve around 40% of the files regardless of the node density. Note that even if the number of nodes increase, network partitioning still always exist due to the nature of vehicle movement patterns considering traffic lights, and therefore there are always some nodes not instantly reachable to the requester.

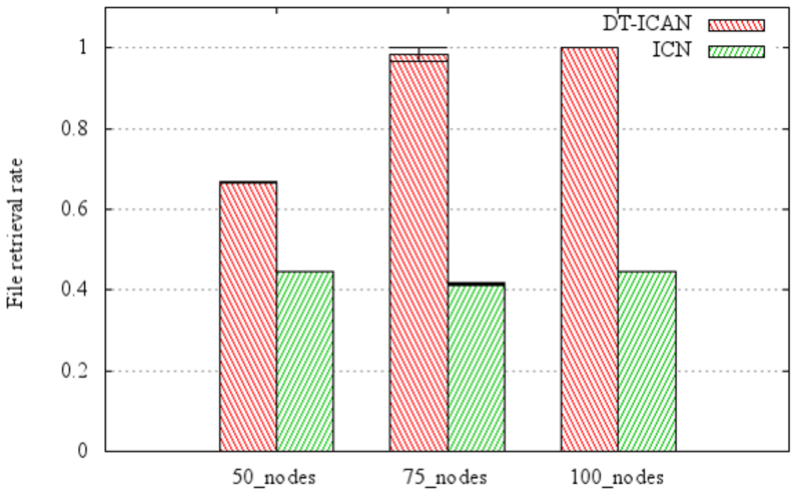

Figure 6.1: Washington DC map: file retrieval rate

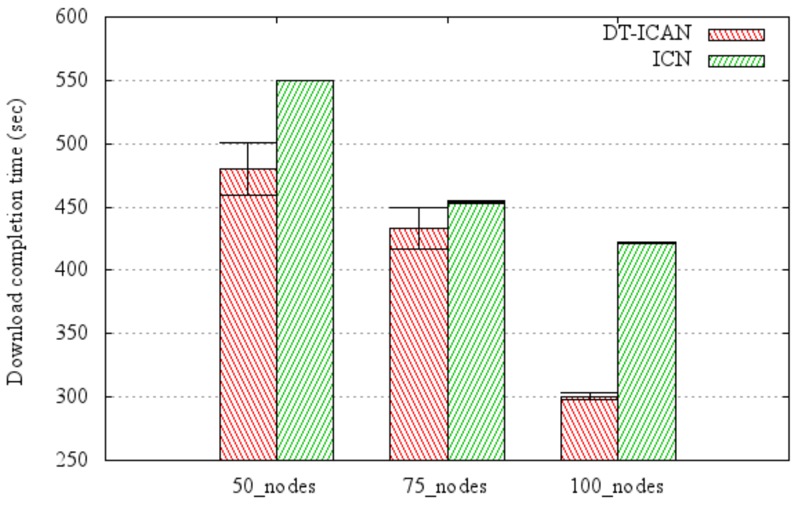

Figure 6.2: Washington DC map: file completion delay

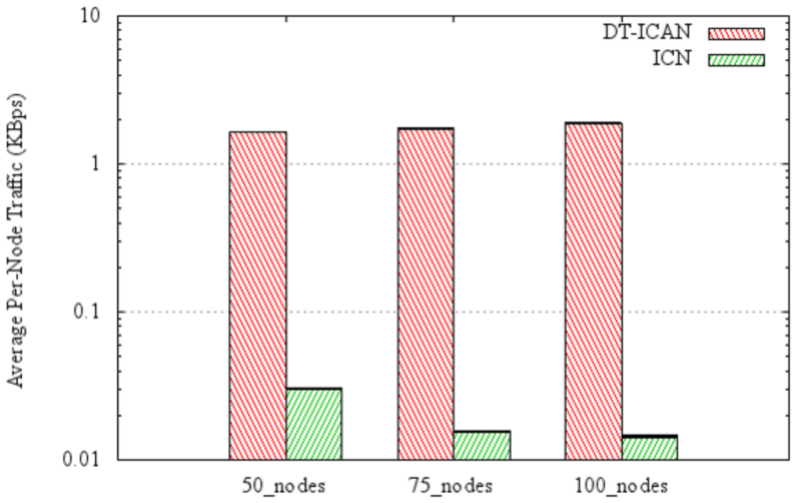

Figure 6.3: Washington DC map: average traffic

We present the completion delay in figure 6.2. As expected, the completion delay for both schemes decrease as the number of nodes increases. The reason is that more caches are available and the chance of node encounters is higher. DT-ICAN achieves up to 30% lower completion delay than that of ICN, as in DT-ICAN multiple caches may request different chunks of the same file simultaneously, leading to better cache diversity than that of ICN. This shows the benefit of data object-level requesting. That is, multiple relays can download different parts of the files for a requester in parallel in DT-ICAN. In ICN, the requests are issued chunk by chunk. Therefore, relays may only request the same chunk(s) at any given time, and hence causes longer delay in a disruptive scenario.

Finally, the average traffic is shown in Figure 6.3. As expected, the performance gain of DT-ICAN comes with the sacrifice in bandwidth consumption. DT-ICAN has constant traffic as the control traffic, i.e. periodic control messages, persists. However, the aggregated traffic is controlled within 10KBps, which is relatively low in the 36MBps channel.

6.1.4.2 San Francisco taxi trace

In this scenario, we use actual mobility traces of taxi cabs in San Francisco downloaded from Crawdad [69]. The simulation runs in a 5700m by 6600m area with 116 taxies. The simulation time is 3600 seconds. Nine data publishers each publishes a 128KB file. One randomly selected data consumer requests the files. All experiments are repeated ten times. In this scenario, the nodes are sparse because only cab movements are recorded in the trace file. This means that the network's period of disconnection is likely longer than its connectivity. However, end to end path still exists in some cases. This trace very well illustrates the case when ICN is deployed as an overlay.

The simulation results are summarized in Table 6.1. Although the network is very sparse, given enough execution time, DT-ICAN is still able to achieve 100% file delivery, double of ICN's file retrieval ratio. The average completion time of DT-ICAN is shorter, as expected, and the average traffic is in the same degree as in the previous experiment. Note that although DT-ICAN outperforms multi-hop retrieval of ICN in general, we still find the merit of multi-hop direct requesting. When an end to end path can be found, the completion time can be dramatically shorter if using multi-hop retrieval. This can be seen by observing the minimum download completion time among all files'. When an end-to-end path exists, ICN is able to complete the transfer within a second, while DT-ICAN still needs more time, i.e. in the degree of tens of seconds, for the epidemic interest propagation. This is due to the fact that DT-ICAN must wait for the node-interest to reach the neighbors of data source. Therefore, we suggest an adaptive approach as proposed in [70], in which the retrieval only switches to DTN mode when the end-to-end connectivity is judged inexistent.

We also explored the parameter space of DT-ICAN in this real-world scenario. The most fundamental parameter in DT-ICAN is the interval for periodic control

	DT-ICAN Mean(Var)	ICN Mean(Var)
File retrieval rate	1 (0)	0.55 (0)
Completion time	821.11 (25.52)	1353.94 (0.61)
Per-node traffic (KBps)	13.43 (2.8)	0.36 (0.02)
Minimum completion time	6.5448 (2.82)	0.2271 (0.03)

Table 6.1: Taxi trace simulation results

messages. In this experiment, we vary these intervals. The completion delay is presented in Figure 6.4. As we can see, the lower completion delay is achieved at interval 2 seconds. Shorter intervals cause slightly higher delay due to the queueing time of the messages. Longer intervals also have higher delay since the data transmissions are triggered more slowly. Due to the limited space, the file retrieval rate and traffic are omitted. DT-ICAN is able to retrieve all files with any settings, and the average per-node traffic remains close to 6KBps for interval larger than 1 second. Therefore, we judge that the 2 second interval is the optimal setting.

6.2 Network Coded DT-ICAN

6.2.1 Network Coding

In disruptive scenarios, given the limited contact duration, the relays are most likely to obtain only partial files. These pieces may be different but some of them are repeated at each relay. Thus, when requesting a data object from multiple relays, the pieces are likely to arrive out of order. In addition, missing pieces make it difficult for a receiver to reliably reconstruct a file. This is so-called **last coupon problem.**

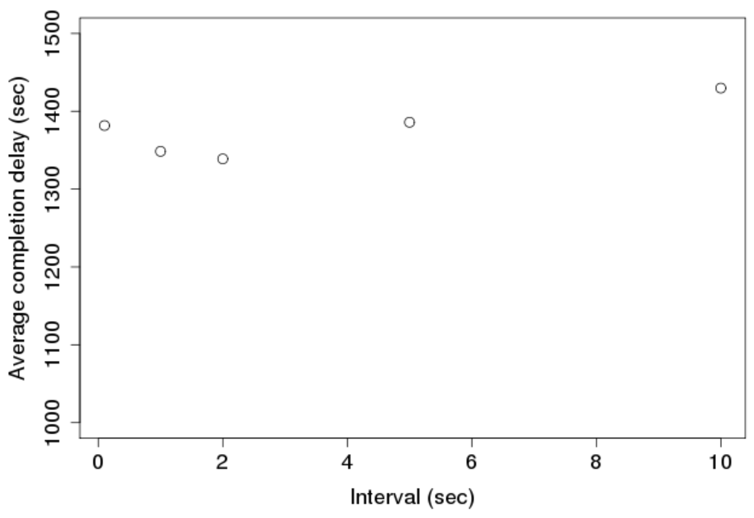

Figure 6.4: Varying control message interval: completion delay

One potential remedy of last coupon problem is network coding. Network coding [71, 72] has been used in MANET [33] for data dissemination to overcome this issue. With network coding, the content originator and the relays encode and re-encode as the available chunks of a file as *coded blocks*, and propagate the coded blocks to the network. The algorithm of network coding is as follows. A source node publishes a file F. In order to disseminate the file in pieces using network coding, the source node first transforms F into a set of m vectors (i.e. chunks) $\mathbf{v_1}, ..., \mathbf{v_m}$ in an n-dimensional vector space over a finite field $\mathbb{GF}_p$ where p is a prime number. These vectors are then linearly combined by drawing from the finite field $\mathbb{GF}_p$ an encoding coefficient $\mathbf{e_i}$ to linearly combine with the vector to create m coded blocks $\mathbf{b_1}, ..., \mathbf{b_m}$. The set of these coefficients then forms the encoding vector $\mathbf{e}$ with $[\mathbf{e_1}, ...\mathbf{e_n}]$. To reconstruct the file, a node simply must receive enough linearly independent coded blocks to be able to perform matrix inversion. First, we take the transpose of the received vectors such that $\mathbf{E^T} = [\mathbf{e^T}_\mathbf{1}, ..., \mathbf{e^T}_\mathbf{n}]$, $\mathbf{B^T} = [\mathbf{b^T}_\mathbf{1}, ..., \mathbf{b^T}_\mathbf{n}]$ and $\mathbf{V^T} = [\mathbf{v^T}_\mathbf{1}, ..., \mathbf{v^T}_\mathbf{n}]$. Then we take $\mathbf{E}^{-1}\mathbf{B}$ which will then reconstruct all the original blocks in the file. Figure 6.5 illustrates how the receiver can fetch linear coded blocks from multiple sources.

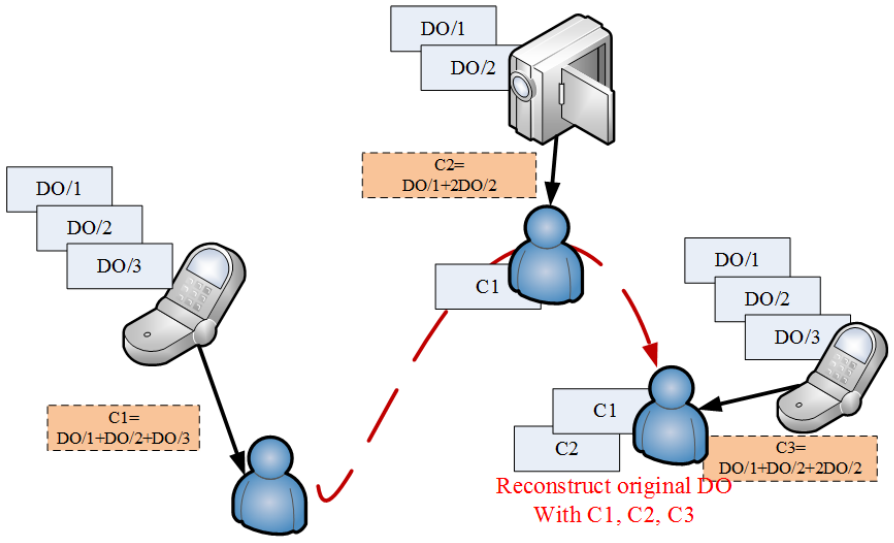

Figure 6.5: A receiver fetches data object from multiple caches using network coding.

The major advantage of this approach is that out-of-order is no longer an issue. The requester can retrieve coded blocks from any node, and reassemble the original file as long as it obtains enough number of linearly independent blocks. As each independently generated coded blocks are likely to be linearly independent, the bandwidth saving in a disruptive scenario due to control overhead and redundant data blocks at relays can be huge. However, network coding exposes security concerns in an ICN due to the possibility of pollution attack. Meanwhile, network coding also requires extra packet overhead to carry the coefficients and extra processing time dedicated to the encoding/decoding process. In the following, we first discuss and evaluate the design options of secure network coding in DT-ICAN. Later, we evaluate the network coded DT-ICAN in smartphone testbed for its performance gain and processing overhead.

6.2.2 Network coding security and design options

Traditional random linear network coding in MANETs [33] forces re-encoding any available chunks at relays even if the cached data objects are partial. This approach is referred as *unrestricted coding* in the following discussion. The benefit of unrestricted coding is its high data dissemination efficiency and resistance to coupon collector problem. However, it is fragile to *pollution attack*. Imagine a malicious node injects invalid coded blocks into the network. With unrestricted coding, these polluted blocks are re-encoded with valid coded blocks by honest intermediate nodes.

In order to control the damage of pollution attack, the direct approach is to require all encoders sign the coded blocks so that if a pollution attack is detected, the malicious node can be identified and blacklisted. Previous network coding studies proposed to use homomorphic signature [73, 74, 75], which is preserved through linear combinations. However, the processing cost of homomorphic signatures is two order of magnitude higher than the encoding cost, making it prohibitive on smaller devices such as smart phones. Since a signature implies that the signer has verified the data object, without using homomorphic signatures, an intermediate node must only sign a coded block if he has received the full data object and verified the integrity of the decoded data object. Since caches may often consist of only partial objects in disruptive environments, unrestricted coding prevents the use of signatures. To this end, we consider two alternative network coding approaches that are secure and feasible:

- *Source only coding*: allows only the publisher to encode and to sign all the coded blocks.
- *Full cache coding*: allows certified intermediate nodes that have fully reassembled the data object to perform re-encoding. The certified intermediate node also signs the regenerated coded blocks in addition to the originator.

There is a tradeoff between the dissemination efficiency and the processing overhead between source only coding and full cache coding. In this section, we (1) evaluate the performance gain of full cache coding over source only coding, and (2) measure how much sacrifices are made for security concern, that is, the performance difference between unrestricted coding and the above two secure alternatives.

6.2.3 Performance And Reliability Analysis

We first analyze the performance of no coding, source only coding, unrestricted coding and full-cache coding. In a disruptive scenario, the intermittency enlarges the waiting time between each blocks and thus higher the chance of multiple full caches before the data object retrieval is complete, the performance of network coding approaches are less predictable by analysis in DTN mode. Thus, we analyze the performance of each approach under a static scenario with random packet losses as a start, and carry out our hypothesis for mobile scenarios later.

Consider the corridor model in Figure 6.6. The origin of the file is the node S. Node R has issued a node-interest. For simplicity, we assume the node-interest contains the interest for only one data object and the node-interest and data are propagated immediately. In addition, we assume pure broadcast without mechanisms such as retransmission and reliable protocols. The node-interest traces three paths by propagation. The model depicts an arbitrary situation when the data object is transmitted through three disjoint "paths", that is, a set of carry-and-forward relays.

6.2.3.1 No Coding

Without network coding, the only way to compensate loss is by duplicate transmissions. We consider the braid scenario in Figure 6.6 as an h hop network since

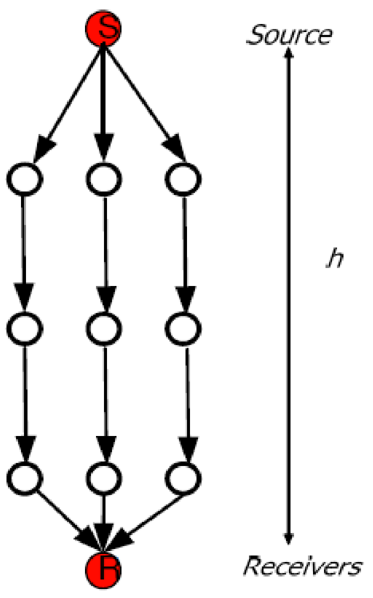

Figure 6.6: 1-3-3-1 corridor model [1]

a packet can be propagated by either of the three relays at each of the h stages and reach the node R in h hops.

Each chunk is triplicated by broadcast. Thus, 3 copies travel along the braid. Suppose each chunk has a chance to be lost in each single-hop transmission under random channel loss. If we define the transmission time of each chunk is T between each hop, the minimum time for N chunks to propagated over h hops is

$$T_b = \max(N, h)T \tag{6.1}$$

Consider a brute-force approach for loss recovery in which all nodes send duplicate chunks in advance. To ensure the perfect delivery in Figure 6.6, suppose the link loss rate is l, each relay must transmit $\lceil \frac{1}{l} \rceil$ duplicates, meaning the bandwidth consumption will always be $\lceil \frac{1}{l} \rceil$ times of the original one and the delay at the receiver must be close to $\lceil \frac{1}{l} \rceil T_b$. One may also consider retransmission upon requests, but this leads to even longer delay as the packet loss can only be detected by timeouts.

6.2.3.2 Source Only Coding

Now suppose the source encodes the blocks. By the principle of linear algebra, the node R is able to reassemble the data object as long as it receives enough number of encoded blocks. In this case, suppose the link loss rate can be estimated, the source S may calculate the number of blocks needed to ensure successful delivery. For example, given the link loss rate l, the probability of successfully deliver a coded block to the node R over h hops on a path i is

$$P_i = (1 - l)^h \tag{6.2}$$

For the braid topology in Figure 6.6, the probability of a block loss on all k paths is

$$l_{all} = (1 - l)^{hk} \tag{6.3}$$

Therefore, the expected number of coded blocks to be transmitted from the source is

$$N_a = \frac{1}{(1 - l)^{hk}} \tag{6.4}$$

Hence, the expected delay of the data object propagation is $N_a T_b$. In other words, source coding has a performance gain as long as $N_a < \frac{1}{l}$, for example, when $l \leq 0.17$ for this particular braid scenario ($hk = 9$).

6.2.3.3 Unrestricted Coding

We now discuss the performance of unrestricted coding. Consider Figure 6.6, if coded blocks are transmitted at the time right when they are received at the relays, intermediate node re-encoding is not useful and the performance gain is equivalent to that of source encoding. In order to benefit from unrestricted coding we must assume multiple blocks are accumulated at the relays and re-encoded to generate new encoded blocks and thus provide better block diversity. In this case, if a block is lost on a strand of the braid, the next coded block will allow recovery. The more blocks accumulated at a node, the more losses we can recover by the intermediate node generated, new linearly independent blocks. In addition, when the blocks are reassembled at the end, the triple redundancy of the three strands also comes to help. In all, in order to accumulate at least two blocks for re-encoding at each hop, the minimum delay of unrestricted coding is $hT + T_b$.

6.2.3.4 Full Cache Coding

Now consider the case of full cache coding. Suppose the blocks are coded at the source and broadcast. As unrestricted coding, full cache coding only improves the performance if the intermediate nodes can re-encode the blocks. Therefore, full cache coding nodes must assemble the data objects first before re-encoding to outperform source only coding. If all blocks must be accumulated before forwarding, the delay can be as high as hNT. However, we are interested in the case where we allow the nodes start forwarding source-encoded blocks before they assembled the data object, and continue with innovative blocks when they receive the full data object. Once the upstream nodes have assembled the files, they can re-encode the blocks and generate as many new coded blocks as necessary to compensate for the lost blocks. The reliability gain of full cache coding, once the cache has already obtained the full file, is as good as the unrestricted coding's, but the minimum delay can be as short as T_b suppose the best case when all transmissions are successful.

The performance gain difference of full cache coding and unrestricted coding comes from the duration when the full data object has not been received by the intermediate nodes, and all intermediate nodes are getting the same set of blocks from upstream. During this time, the performance gain of full cache coding is the same as that of source only coding. In a static scenario where the processing and transmission delay are minimal and no waiting time or source diversity, the performance gain of full cache coding over source only coding depends on the loss probability. For the static braid scenario, if the loss probability is high and assuming the source keeps populating new coded blocks to the network, the chance that some of the caches receive the full data object before the receiver receives enough blocks is high under high loss rate. Therefore, the higher the loss probability is, the larger the performance gain of full cache coding is. As mobility comes in so that the hop by hop transmission time is enlarged, full cache coding has more time and higher chance to accumulate blocks and encode, and therefore will perform more closely to unrestricted coding.

6.2.4 Hypothesis

According to the above analysis, we have the following hypotheses for mobile and intermittent scenarios:

1. **Full cache coding provides higher performance gain than source only coding**. Suppose there is only one originator in the network for the particular file. Once the blocks have been reassembled by caches, a receiver may receive blocks from multiple caches, as argued in 6.2.3.4. On the other hand, if there are multiple originators for the same file, the block diversity provided by multiple originators may shorten the performance gain difference between source only coding and full cache coding.

2. **Full cache coding provides comparable performance to unrestricted coding in an intermittent network**. As the network is intermittent, the blocks received by each partial cache are more likely to be different and thus linearly independent by nature. Meanwhile, consider the case that after a sufficient time of operations, when there are already multiple full caches in the intermittent scenario, the encoded blocks provided by them can offer sufficient diversity even if the partial caches do not re-encode. In this case, the performance of full cache coding can be comparable to that of unrestricted

coding. Considering full cache coding has the advantage of pollution attack protection, it is a better coding approach for DT-ICAN if the performance gain is comparable to that of unrestricted coding.

6.2.5 Simulation

We evaluate the performance of unrestricted coding, full-cache coding, source-only coding, and no coding by simulation using DT-ICANSIM [1], which is implemented in Qualnet 6.1.

6.2.5.1 Static scenario

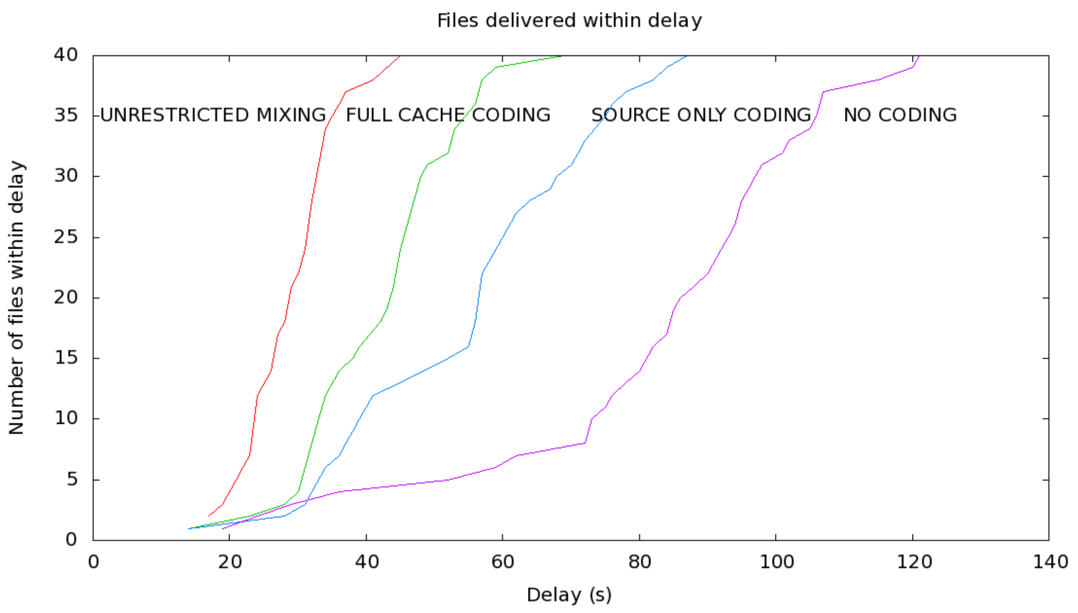

Figure 6.7: Simulation results: corridor model with 30% loss probability

We first examine a static, lossy scenario using the corridor model in Figure 6.6. The loss rate is 30%. The MAC layer uses IEEE 802.11a and data rate 54Mbps. The transmission range is about 70 meters. In this scenario, we have one publisher and one subscriber. For simplicity, we evaluate a single file transmission. Note

[1]https://github.com/uclanrl/dt-icansim

that our results can be generalized to multiple files as our technique is not bound to number of files. For no coding, we consider the case without the brute-force loss recovery in the simulation.

The results are shown in Figure 6.7. We observe that as expected, the performance gain of the three network coding approaches follows *Unrestricted coding* > *Full cache coding* > *Source only coding.* This matches our analysis as the unrestricted coding has the advantage of intermediate node re-encoding before the full file is received by the intermediate nodes. Full cache coding performs better than source only coding due to the ability to add diversity after the full file is obtained by the intermediate caches under high loss probability.

6.2.5.2 Mobile scenario

We next study a mobile scenario using random waypoint model. This scenario consists of 10 nodes, including three publishers and seven receivers. The territory size is 1000 by 1000 meters. The parameters of the random waypoint model are minimum speed of 1 m/s, maximum speed of 3 m/s, pause time of 1 second, and a total duration of 10 minutes.

The results for our mobile scenario is presented in Figure 6.8. Despite the similar performance, looking into the slightly different performance gain, among the three network coding approaches, full cache coding does the best. More data objects are delivered within 20 seconds by full cache coding and source only coding than unrestricted coding. The reason is that as unrestricted coding requires accumulating innovative blocks before starting forwarding, in the intermittent scenario, the delay can become longer due to the time for accumulation, and the other two outperform when the network is more connected by simply forwarding single blocks at some relays. When it comes to the time the network is more disruptive, full cache coding outperforms source only coding successfully deliver

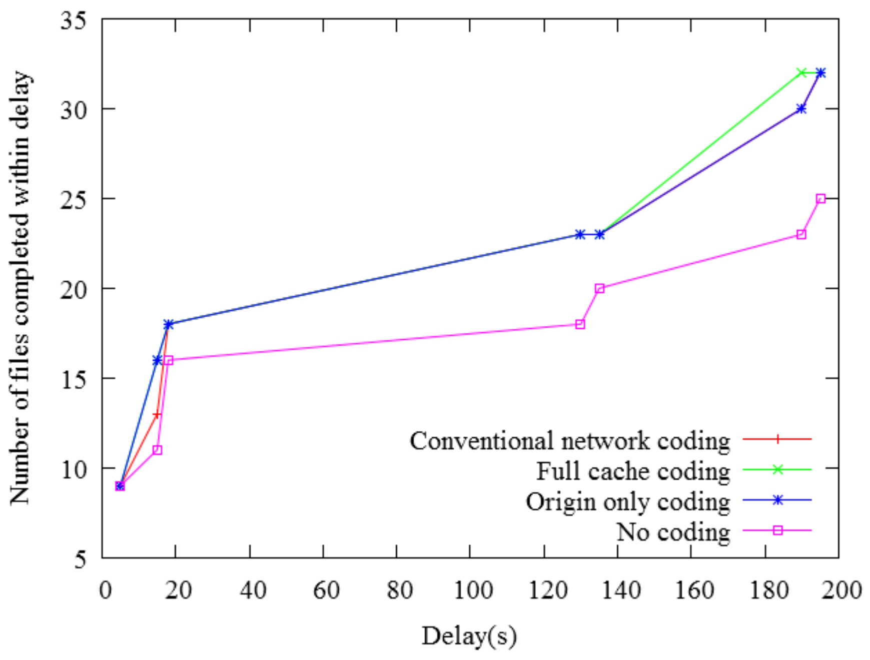

Figure 6.8: Simulation results: random waypoint

more data objects with higher delay. In all, the results show that the intermittence and multi-source have offered enough diversity for full cache coding and source only coding, and thus the performance gain is comparable to that of unrestricted coding. Given the fact that full cache coding naturally creates more sources and its resistance to pollution attacks, full cache coding is an ideal approach for coding in DT-ICAN.

6.3 Testbed evaluation

Given the above results, we select full cache coding as our network coding approach. Even if the re-encoding is restricted to full caches, coding requires extra packet overhead to carry the coefficients and extra processing time dedicated to the encoding/decoding process. In some cases, randomizing the order of the chunks may provide enough diversity without excessive processing. In this section, we explore this possibility in practical military DTN scenario and explore the tradeoff between the performance gain and processing overhead.

6.3.1 Emulation

6.3.1.1 Emulation settings

In order to better understand the tradeoff between the performance gain and processing and transmission overhead, we use emulation for this study. The network emulation is done using EMANE 802.11 [76], which includes a path loss model and interface to the real-time Common Open Resource Emulator, CORE 4.3 [77], thus allowing us to run realistic emulation experiments on real hardware.

We first compare the performance of source coding and full cache network coding and the original randomized chunk delivery mechanism of DT-ICAN, denoted by no coding thereafter, in which when a request for data object is received at a data source or relay, data chunks are sent by a randomly permuted order. The metrics evaluated include delivery fraction and average delay. The delivery fraction is calculated as the fraction of the total number of data objects received to the expected number of data objects that should be received in the system. The delivery fraction is computed system-wise. The same data object requested by multiple requesters are counted multiple times. That is, if two requesters ask for the same data objects, we consider the expected number of data objects to be received as two. The average delay is defined as the interval between the data object is requested to the data object is received for each requester.

6.3.1.2 Data Mule Scenario

We first test the performance of full cache coding, source only coding, and no coding in a data mule scenario (Figure 6.9). The scenario is inspired by a practical military VANET scenario in which two squads (16 and 12 members each) exchange information through mobile vehicles (data mules). In this scenario, we assume

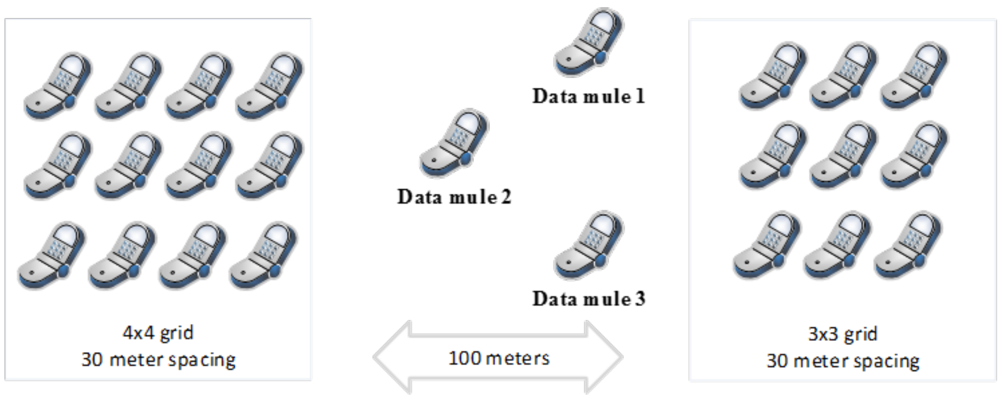

Figure 6.9: Data mule scenario

three data mule nodes ferry data objects between two large groups (grids) of nodes. Each node within the grid is 30 meters apart from each other. In each grid, there are three publishers, and each publishes a 512K data object. We assume a data dissemination application and thus the total of six data objects are subscribed by all nodes. The data mule speed is 1.4 m/s. Each data mule is out of range of other data mules and stays within range to each grid for 20 seconds. The experiment duration is 600 seconds. This model allows us to evaluate the effects of largely connected groups interconnected by a short-term and low-bandwidth connection. We expect to see the coupon collector problem slowing down cross pollination file reconstruction between the two groups.

	Delivery Fraction	Average Delay (seconds)	Std Deviation
No Coding	58.3%	68.5095	42.2422
Source only coding	78.3%	66.7729	47.8758
Full cache coding	100%	70.8891	43.4677

Table 6.2: Data mule scenario results

In Table 6.2, we observe that full cache coding delivers close to twice the number of files of no coding and also outperforms source only coding. No coding has low delivery fraction due to the coupon collector problem. The data mules provide only short contact and thus insufficient bandwidth for complete data object deliveries via each data mule at each contact. Full cache coding ensures higher diversity of blocks and consequently better bandwidth utilization, while with no coding the data mules are more likely to carry redundant chunks, even with the randomization. The per-data object delay of source coding is shorter than full cache coding due to its low processing overhead. However, the delivery fraction of source only coding is still lower as the existence of full caches can further improve total bandwidth utilization in this scenario.

We ran several variations of the study to examine the effects of increasing the number of data mules and the duration each data mule stayed connected with each squad. Interestingly, even varying the number of data mules or the stay duration while still maintaining the scenario duration of 600 seconds, no coding still cannot achieve 100% delivery fraction. Therefore, we judge that low-bandwidth interconnects due to mobility and packet loss require the use of intermediate node coding.

6.3.1.3 Search Patrol Scenario

Consider the search patrol model (Figure 6.10), where a search patrol visits rendezvous points and shares reconnaissance and updates info. In the search patrol scenario, there are 2 squads with 14 members each. Each squad is composed of subgroups (2 subgroups of 7 members) to avoid the enemy wiping out the entire squad with a single shot. Intra-squad communication using 802.11 occurs periodically at each rendezvous (central server, hot spot, and red box).

Both the central server at rendezvous 1 and rendezvous 2 publish a 1MB file to squad 1 and squad 2, respectively. Additionally, squad 1 and squad 2 upload

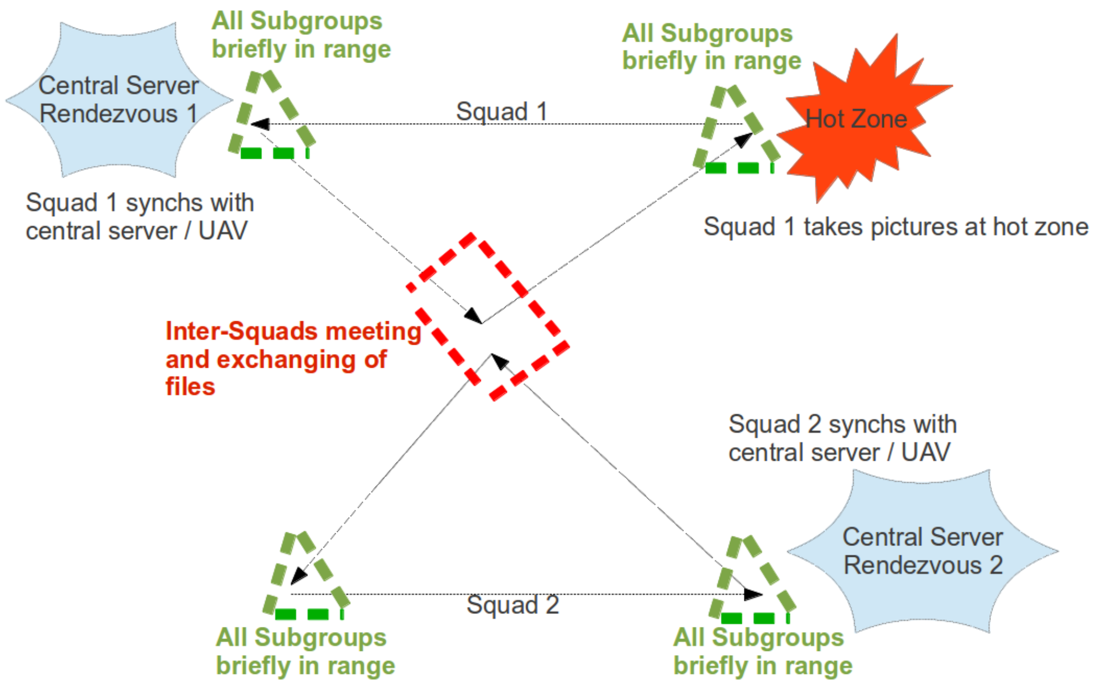
All Subgroups briefly in range
All Subgroups briefly in range
Central Server Rendezvous 1
Squad 1
Hot Zone
Squad 1 synchs with central server / UAV
Squad 1 takes pictures at hot zone
Inter-Squads meeting and exchanging of files
Squad 2 synchs with central server / UAV
Central Server Rendezvous 2
Squad 2
All Subgroups briefly in range
All Subgroups briefly in range

Figure 6.10: Search patrol scenario: 2 squads (composed of 3 sub-squads) walk in a triangle pattern between 3 rendezvous points.

their collected data objects to server 1 and server 2, respectively. Each scout in each subgroup takes a 512KB picture at the hot zone and performs intra-subgroup sharing between and within each rendezvous. This results in a total of 4 data objects published and shared to a total of 30 subscribers. Intra-squad sharing occurs only at each rendezvous point. Inter-squad file sharing occurs in the middle red box. Due to intermittence, each node is likely to holds only partial data object during the first few cycles. The partial caches consist of the files from central server 1, central server 2, and pictures taken at the hot zone. The cycling continues for 600 seconds.

	Delivery Fraction	Average Delay (seconds)	Std Deviation
No Coding	82.9%	245.888	182.097
Source Coding	85.3%	186.159	186.12
Full cache coding	100%	106.928	86.0384

Table 6.3: Search patrol emulation scenario with 20 second rendezvous.

	Delivery Fraction	Average Delay (seconds)	Std Deviation
No Coding	100%	130.086	115.55
Source Coding	100%	88.5397	81.7264
Full cache coding	100%	96.1285	85.3548

Table 6.4: Search patrol emulation scenario with 45 second rendezvous.

We compare a short duration stay of 20 seconds to a longer duration stay of 45 seconds as seen in Table 6.3 and Table 6.4 respectively. With higher intermittency degree (20 second rendezvous time), only full cache coding is able to achieve 100% delivery due to its ability to overcome the coupon collector problem. However, in extended connectivity (45 second rendezvous time), all approaches can achieves 100% delivery. Meanwhile, source only coding achieves lowest average delay in this case. This scenario demonstrates that the variation of environmental parameters results in two different outcomes: namely, one where full cache coding is required and one where full cache coding may backfire due to its higher processing overhead. In all cases, full cache coding is sufficient to deliver all data objects if a low delay is not required. Therefore, we judge that full cache coding is a useful technique in disruptive scenarios. In addition, we observe the potential of enhancing network coding performance by restricting intermediate node coding under assistance of mobility pattern awareness. We list this potential enhancement as a possible future work.

CHAPTER 7

Conclusion

We present ICAN, an ICN-based ad-hoc networking architecture. ICAN generalizes the well-known named data principle of ICN. In ICAN, names are assigned not only to data, but also to physical entities such as hosts and geographic areas. Being a superset of existing ICN and host-based networks, ICAN provides built-in network condition and application context awareness together with the broad network entity definitions, thus enabling the dynamic adoption of network operations like routing and caching mechanisms.

ICAN achieves high efficiency, flexibility and backward-compatibility. Unlike current ICN proposals, ICAN supports push transport and context-aware forwarding and caching. This work aims at improving the ad-hoc ICN efficiency by utilizing the context awareness in all aspects. We extensively study the design options of VANET ICN system by practical large-scale simulation. We also implement a proof of concept prototype on our Android testbed. We conclude that pervasive caching is critical and source-only routing is sufficient in VANETs. With this insight, we propose context aware content discovery mechanism for ICAN. In addition to the extensive study in urban VANET scenarios, we further explore efficient data propagation for the DTN mode of ICAN. We utilize secure network coding technique to speed up the data retrieval in intermittent scenarios. In our smartphone emulation testbed, we verify that network coding is helpful even after accounting for its additional processing overhead.

References

[1] S. Y. Oh, M. Gerla, and A. Tiwari, "Robust manet routing using adaptive path redundancy and coding," in *Proceedings of the First international conference on COMmunication Systems And NETworks*, COMSNETS'09, (Piscataway, NJ, USA), pp. 224–233, IEEE Press, 2009.

[2] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 163–168, IEEE, 2011.

[3] A. Brown, "Google's autonomous car applies lessons learned from driverless races," *Mechanical Engineering*, vol. 133, no. 2, p. 31, 2011.

[4] U. Lee, B. Zhou, M. Gerla, E. Magistretti, P. Bellavista, and A. Corradi, "Mobeyes: smart mobs for urban monitoring with a vehicular sensor network," *Wireless Communications, IEEE*, vol. 13, no. 5, pp. 52–57, 2006.

[5] U. Lee, E. Magistretti, M. Gerla, P. Bellavista, and A. Corradi, "Dissemination and harvesting of urban data using vehicular sensing platforms," *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 2, pp. 882–901, 2009.

[6] A. Nandan, S. Das, G. Pau, M. Gerla, and M. Sanadidi, "Co-operative downloading in vehicular ad-hoc wireless networks," in *Wireless On-demand Network Systems and Services, 2005. WONS 2005. Second Annual Conference on*, pp. 32–41, IEEE, 2005.

[7] V. Jacobson, M. Mosko, D. Smetters, and J. Garcia-Luna-Aceves, "Content-centric networking," *Whitepaper, Palo Alto Research Center*, pp. 2–4, 2007.

[8] S. Biswas and R. Morris, "Opportunistic routing in multi-hop wireless networks," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 69–74, 2004.

[9] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, "Scalable routing strategies for ad hoc wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 17, no. 8, pp. 1369–1379, 1999.

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

[11] C. Dannewitz, "Netinf: An information-centric design for the future internet," in *Proc. 3rd GI/ITG KuVS Workshop on The Future Internet*, 2009.

[12] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "Mdht: a hierarchical name resolution service for information-centric networks," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pp. 7–12, ACM, 2011.

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

[14] J. L. D. K. Myeong-Wuk and J. B.-J. Lee, "Proxy-based mobility management scheme in mobile content centric networking (ccn) environments," Consumer Electronics (ICCE), 2011 IEEE International Conference on, 2011.

[15] S. Oh, D. Lau, and M. Gerla, "Content centric networking in tactical and emergency manets," in *Wireless Days (WD), 2010 IFIP*, pp. 1–5, 2010.

[16] M. Varvello, I. Rimac, U. Lee, L. Greenwald, and V. Hilt, "On the design of content-centric manets," in *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*, pp. 1–8, IEEE, 2011.

[17] M. Meisel, V. Pappas, and L. Zhang, "Listen first, broadcast later: Topology-agnostic forwarding under high dynamics," in *Annual Conference of International Technology Alliance in Network and Information Science*, p. 8, 2010.

[18] M. Amadeo, A. Molinaro, and G. Ruggeri, "E-chanet: Routing, forwarding and transport in information-centric multihop wireless networks," *Computer Communications*, 2013.

[19] M. Amadeo, C. Campolo, and A. Molinaro, "Crown: Content-centric networking in vehicular ad hoc networks," 2012.

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

[21] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA'99. Second IEEE Workshop on*, pp. 90–100, IEEE, 1999.

[22] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Kluwer International Series in Engineering and Computer Science*, pp. 153–179, 1996.

[23] G. Pei, M. Gerla, and T.-W. Chen, "Fisheye state routing: A routing scheme for ad hoc wireless networks," in *Communications, 2000. ICC 2000. 2000 IEEE International Conference on*, vol. 1, pp. 70–74, IEEE, 2000.

[24] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, L. Viennot, *et al.*, "Optimized link state routing protocol (olsr)," 2003.

[25] S. Keshav, "An engineering approach to computer networking: Atm networks, the internet, and the telephone network," *Reading MA*, vol. 11997, 1997.

[26] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," in *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pp. 3866–3875, IEEE, 2002.

[27] Y.-B. Ko and N. H. Vaidya, "Location-aided routing (lar) in mobile ad hoc networks," *Wireless Networks*, vol. 6, no. 4, pp. 307–321, 2000.

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

[29] A. Vahdat, D. Becker, *et al.*, "Epidemic routing for partially connected ad hoc networks," tech. rep., Technical Report CS-200006, Duke University, 2000.

[30] B. Burns, O. Brock, and B. N. Levine, "Mora routing and capacity building in disruption-tolerant networks," *Ad hoc networks*, vol. 6, no. 4, pp. 600–620, 2008.

[31] R. Ramanathan, R. Hansen, P. Basu, R. Rosales-Hain, and R. Krishnan, "Prioritized epidemic routing for opportunistic networks," in *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking*, pp. 62–66, ACM, 2007.

[32] C. Boldrini, M. Conti, and A. Passarella, "Impact of social mobility on routing protocols for opportunistic networks," in *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pp. 1–6, IEEE, 2007.

[33] U. Lee, J.-S. Park, J. Yeh, G. Pau, and M. Gerla, "Code torrent: content distribution using network coding in vanet," in *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, MobiShare '06, (New York, NY, USA), pp. 1–5, ACM, 2006.

[34] J.-S. Park, M. Gerla, D. Lun, Y. Yi, and M. Medard, "Codecast: a network-coding-based ad hoc multicast protocol," *Wireless Communications, IEEE*, vol. 13, no. 5, pp. 76–81, 2006.

[35] M.-J. Montpetit, C. Westphal, and D. Trossen, "Network coding meets information-centric networking: an architectural case for information dispersion through native network coding," in *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications*, NoM '12, (New York, NY, USA), pp. 31–36, ACM, 2012.

[36] Q. Wu, Z. Li, and G. Xie, "Codingcache: multipath-aware ccn cache with network coding," in *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, ICN '13, (New York, NY, USA), pp. 41–42, ACM, 2013.

[37] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, *Trading structure for randomness in wireless opportunistic routing*, vol. 37. ACM, 2007.

[38] C.-C. Chen, G. Tahasildar, Y.-T. Yu, J.-S. Park, M. Gerla, and M. Sanadidi, "Codemp: Network coded multipath to support tcp in disruptive manets," in *Mobile Adhoc and Sensor Systems (MASS), 2012 IEEE 9th International Conference on*, pp. 209–217, IEEE, 2012.

[39] Y. Lin, B. Li, and B. Liang, "Efficient network coded data transmissions in disruption tolerant networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, IEEE, 2008.

[40] M. Chuah, P. Yang, S. Roy, and B. Sheng, "Performance evaluation of dissemination schemes for coded packets in heterogeneous sparse ad hoc networks.," *Ad Hoc And Sensor Wireless Networks*, vol. 15, no. 2-4, pp. 151–181, 2012.

[41] P. Mundur and M. Seligman, "Delay tolerant network routing: Beyond epidemic routing," in *Wireless Pervasive Computing, 2008. ISWPC 2008. 3rd International Symposium on*, pp. 550–553, IEEE, 2008.

[42] E. C. de Oliveira and C. V. de Albuquerque, "Nectar: a dtn routing protocol based on neighborhood contact history," in *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 40–46, ACM, 2009.

[43] C. Tsilopoulos and G. Xylomenos, "Supporting diverse traffic types in information centric networks," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pp. 13–18, ACM, 2011.

[44] Z. Zhu, S. Wang, X. Yang, V. Jacobson, and L. Zhang, "Act: audio conference tool over named data networking," in *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pp. 68–73, ACM, 2011.

[45] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (xml)," *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210*, 1998.

[46] J. Scott, J. Crowcroft, P. Hui, C. Diot, *et al.*, "Haggle: A networking architecture designed around mobile users," in *WONS 2006: Third Annual Conference on Wireless On-demand Network Systems and Services*, pp. 78–86, 2006.

[47] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[48] U. Lee, J. Lee, J.-S. Park, and M. Gerla, "Fleanet: A virtual market place on vehicular networks," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 1, pp. 344–355, 2010.

[49] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," *SIGCOMM Comput. Commun. Rev.*, vol. 43, pp. 147–158, Aug. 2013.

[50] C. L. Miller, "Tiger/line files technical documentation. ua 2000. us department of commerce, geography division, us census bureau."

[51] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "Sumo (simulation of urban mobility)," in *Proc. of the 4th Middle East Symposium on Simulation and Modelling*, pp. 183–187, 2002.

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

[53] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 1, pp. 151–160, 1998.

[54] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: evidence and possible causes," *Networking, IEEE/ACM Transactions on*, vol. 5, no. 6, pp. 835–846, 1997.

[55] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, pp. 126–134, IEEE, 1999.

[56] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," *Relatório técnico, Telecom ParisTech*, 2011.

[57] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking: seeing the forest for the trees," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, p. 1, ACM, 2011.

[58] L. Guo, E. Tan, S. Chen, Z. Xiao, and X. Zhang, "Does internet media traffic really follow zipf-like distribution?," in *SIGMETRICS*, vol. 7, pp. 359–360, 2007.

[59] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[60] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking (TON)*, vol. 8, no. 3, pp. 281–293, 2000.

[61] P. Gu, J. Wang, and H. Cai, "Asap: An advertisement-based search algorithm for unstructured peer-to-peer systems," in *Parallel Processing, 2007. ICPP 2007. International Conference on*, pp. 8–8, IEEE, 2007.

[62] P. Boldi and S. Vigna, "Mutable strings in java: design, implementation and lightweight text-search algorithms," *Science of Computer Programming*, vol. 54, no. 1, pp. 3–23, 2005.

[63] P. S. Almeida, C. Baquero, N. Preguiça, and D. Hutchison, "Scalable bloom filters," *Information Processing Letters*, vol. 101, no. 6, pp. 255–261, 2007.

[64] Y.-T. Yu, T. Punihaole, M. Gerla, and M. Sanadidi, "Content routing in the vehicle cloud," in *MILITARY COMMUNICATIONS CONFERENCE, 2012-MILCOM 2012*, pp. 1–6, IEEE, 2012.

[65] M. Grossglauser and M. Vetterli, "Locating mobile nodes with ease: learning efficient routes from encounter histories alone," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. 3, pp. 457–469, 2006.

[66] N. Sarafijanovic-Djukic and M. Grossglauser, "Last encounter routing under random waypoint mobility," in *NETWORKING 2004. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications*, pp. 974–988, Springer, 2004.

[67] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "Ght: a geographic hash table for data-centric storage," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 78–87, ACM, 2002.

[68] J. Härri, F. Filali, C. Bonnet, and M. Fiore, “Vanetmobisim: generating realistic mobility patterns for vanets,” in *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, pp. 96–97, ACM, 2006.

[69] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, “CRAWDAD data set epfl/mobility (v. 2009-02-24).” Downloaded from http://crawdad.org/epfl/mobility/, Feb. 2009.

[70] Y.-T. Yu, C. Tandiono, X. Li, Y. Lu, M. Sanadidi, and M. Gerla, “Ican: Information-centric context-aware ad-hoc network,”

[71] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, “A random linear network coding approach to multicast,” *Information Theory, IEEE Transactions on*, vol. 52, no. 10, pp. 4413–4430, 2006.

[72] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, “Network coding: an instant primer,” *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 63–68, 2006.

[73] S.-H. Lee, M. Gerla, H. Krawczyk, K.-W. Lee, and E. Quaglia, “Performance evaluation of secure network coding using homomorphic signature,” in *Network Coding (NetCod), 2011 International Symposium on*, pp. 1–6, 2011.

[74] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin, “Secure network coding over the integers,” in *Proceedings of the 13th International Conference on Practice and Theory in Public Key Cryptography*, PKC’10, (Berlin, Heidelberg), pp. 142–160, Springer-Verlag, 2010.

[75] D. Boneh, D. Freeman, J. Katz, and B. Waters, “Signing a linear subspace: Signature schemes for network coding,” in *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography: PKC ’09*, Irvine, (Berlin, Heidelberg), pp. 68–87, Springer-Verlag, 2009.

[76] “Emane.” `http://cs.itd.nrl.navy.mil/work/emane/`, 2013. [Online; accessed 2-April-2013].

[77] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, “Core: A real-time network emulator,” in *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pp. 1–7, IEEE, 2008.