FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Cooperative Content Dissemination on Vehicle Networks

Diogo Recharte

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor at FEUP: Ana Aguiar

Supervisor at Veniam: Henrique Cabral

June 25, 2018

Abstract

Vehicular networks have seen great advancements over the last few years, mostly due to the increased eagerness for smart and autonomous vehicles that motivate hefty investments by the automotive industry.

The absence of a timely and cost-effective way to perform over-the-air (OTA) software updates is contributing to defer the deployment of large fleets of connected vehicles. There is a high cost associated with transmitting data over cellular networks and it cannot be expected that every vehicle has access to a station or depot with adequate connectivity where it can get the awaited data cheaply nor that this solution happens timely enough.

With this in mind, this thesis presents the design and implementation of a cooperative content dissemination protocol that takes advantage of Vehicle-to-Vehicle (V2V) communication links to distribute data across a network with reduced costs. To lessen the effects of short connection duration, the content to disseminate is divided into small chunks. Interested nodes periodically query their neighbors about chunk availability and, upon receiving replies, decide which specific chunk will be exchanged. This work bases the design decisions in real data from Veniam's deployed network in Porto, such as contact statistics and throughput measurements. Moreover, this work is complemented with a performance analysis of the protocol on a deployed network of 17 vehicles and 5 roadside units (RSUs).

The results show that the dissemination time is significantly sped up by V2V cooperation and that is heavily influenced by vehicle mobility. Also, similar results can be achieved in an infrastructure-less network, by allowing some vehicles to become the original seeders of the content. Moreover, results show that the chunk size and decision algorithm can have a considerable impact in the performance of the protocol.

Resumo

As redes veiculares têm sido alvo de grandes avanços nos últimos anos, sobretudo devido ao crescente interesse por veículos inteligentes e autónomos que motiva investimentos avultados por parte da indústria automóvel.

A inexistência de uma forma oportuna e económica de executar atualizações de software over-the-air (OTA) está a contribuir para o adiar do lançamento de grandes frotas de veículos inteligentes. O custo associado à transmissão de dados através de redes celulares é muito elevado e não se pode garantir que cada veículo tenha acesso a uma estação ou estacionamento com conectividade adequada em tempo útil, onde possa obter os dados esperados.

Com base nestas premissas, esta tese apresenta a concepção e implementação de um protocolo cooperativo de disseminação de conteúdos que aproveita as ligações veículo-a-veículo (V2V) para assegurar uma distribuição de dados pela rede, com custos reduzidos. De forma a reduzir os efeitos das conexões de curta duração, o conteúdo a disseminar é dividido em pequenos blocos (chunks). Os nós interessados questionam periodicamente os seus vizinhos acerca dos chunks que estes possuem e, quando recebem respostas, tomam a decisão de qual será o chunk a ser transferido. Este trabalho fundamenta todas as decisões de desenho do protocolo em dados reais provenientes da rede de produção da Veniam no Porto, tais como estatísticas de contactos e medidas de throughput. Além disso, este trabalho é complementado e suportado com uma análise do desempenho do protocolo numa rede de 17 veículos e 5 roadside units (RSUs).

Os resultados mostram que a cooperação V2V reduz consideravelmente o tempo de disseminação e que este é muito influenciado pela mobilidade dos veículos. Além disso, resultados semelhantes poderão ser obtidos ao permitir que alguns veículos se tornem os seeders originais do conteúdo. Os resultados também mostram que o tamanho dos chunks e o algoritmo de decisão podem ter um impacto considerável na performance do protocolo.

Contents

List of Figures

List of Tables

Acronyms

AP	Access Point
API	Application Programming Interface
AODV	Ad Hoc On Demand Distance Vector
CCH	Control Channel
CLI	Command Line Interface
CRC	Cyclic Redundancy Check
DDT	Distance Defer Transfer
DSR	Dynamic Source Routing
DSCF	Directional Store-Carry-Forward
DSRC	Dedicated Short Range Communication
DTN	Delay-Tolerant Network
FDMA	Frequency-Division Multiple Access
FFRDV	Fastest-Ferry Routing in DTN-enabled Vehicular Ad Hoc Networks
FIFO	First-In First-Out
IEEE	Institute of Electrical and Electronics Engineers
IPC	Inter-Process Communication
ITS	Inteligent Transportation System
JSON	JavaScript Object Notation
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
MTU	Maximum Transmission Unit
OBU	On Board Unit
OFDM	Orthogonal Frequency-Division Multiplexing
OLSR	Optimized Link State Routing
OTA	Over-The-Air
PHY	Physical
ROD	Road Oriented Dissemination
RSU	Roadside Unit
RTT	Round-Trip Time
SCH	Service Channel
SPAWN	Swarming Protocol For Vehicular Ad Hoc Wireless Networks
TDMA	Time-Division Multiple Access
TTL	Time To Live
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VANET	Vehicular Ad Hoc Network
VDTN	Vehicular Delay Tolerant Network
WAVE	Wireless Access in Vehicular Environment
WSA	WAVE Service Advertisement
WSMP	WAVE Short Message Protocol

Chapter 1

Introduction

Throughout the years, technology has become present everywhere and the internet of things is becoming a reality where everyone and everything has an internet connection. Vehicular ad hoc networks (VANETs) are seen as one of the most influential technologies for improving road safety and building intelligent transportation systems (ITSs) [1]. In fact, it has been an area of avid research by industries, academic institutions and even governments around the world [2].

Soon every personal and public transportation vehicle will be part of a mesh network where they continuously communicate with each other and the cloud. Smarter vehicles and embedded infotainment systems rely on constant communication with adjacent vehicles to improve safety and drive the need to move terabytes of data to and from the cloud to provide relevant content to the vehicle operation as well as the comfort of the passengers. Moving such large amounts of data over cellular networks is prohibitively expensive, especially when scaling up the network. Dedicated Short-Range Communication (DSRC) technology defines several standards for wireless Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication. Even though DSRC and the series of standards for Wireless Access in Vehicular Environment (WAVE) bring major improvements in connection establishment and maintenance, VANETs, by nature, present a volatile environment. As vehicles move in and out of range from each other, connections are established and lost, and for this reason, it is unrealistic to assume that large sets of data can be exchanged during the duration of a connection. Vehicular Delay Tolerant Network (VDTN) overlays can mitigate the adverse impact of sparse and/or intermittent connectivity on data transmission by making use of relays and store-carry-forward mechanisms to ensure that messages can be delivered to the destination without a synchronous end-to-end connection.

1.1 Veniam

One of the largest networks of connected vehicles in the world, Veniam's VANET in Porto has been at the forefront of this industry with a fleet of over 400 public buses, garbage collection trucks and municipality vehicles equipped with on board units (OBUs) that provided wireless connectivity. Veniam's platform takes advantage of multiple network interfaces and technologies

such as DSRC, Wi-Fi and LTE to build an integrated vehicle ecosystem. Coupled with a versatile network management tool that delivers mobility and seamless handovers between the different technologies, it ensures connectivity through the interface that best caters to the immediate situation. It also implements local data management that decides which data should be offloaded in real time and which data can be stored to be sent later in a delay tolerant manner.

The fleet in Porto is supported by an infrastructure of over 40 roadside units (RSUs) that feature a reliable wired backhaul and are otherwise similar to the vehicle OBUs. The relatively small downtown Porto area holds a good density of 4 RSU/km^2 while the remaining metropolitan area has a density of 0.6 RSU/km^2. The deployment of RSUs can be expensive upfront but the use of an infrastructure to support communications reduces daily costs. In fact, in the last 5 years in downtown Porto, the use of vehicle-to-infrastructure (V2I) communications to offload data to the cloud has yielded 48% cost savings when compared to a 4G only solution.

Currently, Veniam's network enables delay-tolerant communication in the uplink but not in the downlink. This leads to content being downloaded through the interface available at the given moment, which is often the expensive cellular connection. Moreover, automotive manufacturers are in need of cost effective ways to disseminate delay tolerant content, as the ability to perform over-the-air (OTA) software updates is imperative to ensure the longevity of any product. All things considered, Veniam provides a unique environment with timely problems that have not yet been dealt with and the possibility to experimentally evaluate solutions in a real-world environment.

1.2 Objectives and Approach

This work focuses on solving a popular content distribution (PCD) problem, where most or even all nodes in a VANET are interested in downloading a given content. Using cellular connections to distribute content over large fleets can lead to exorbitant expenses, due to the high cost of cellular data. VANETs supported by an infrastructure with RSUs allow for the use of DSRC technology to establish V2I links and distribute the content in a cheap way. However, the infrastructure deployment is expensive and RSUs are often scarce. The scarcity of these units and the speed of the vehicles lead to V2I contacts having a short duration which often does not allow for OBUs to completely download the content.

In this work, a cooperative data dissemination protocol is proposed. By taking advantage of V2V communication links, a peer-to-peer (P2P) network overlay is formed and a cooperative file sharing system is created. Then, vehicle mobility spreads the content over a large area, and consequently, dissemination time is reduced.

1.3 Structure of the Document

After this brief introduction, in which some basic concepts and topics that serve as motivation for this work were presented, a deeper analysis of vehicle networks and related background work is presented in Chapter 2. In Chapter 3, protocol design decisions and implementation details are

explained. In Chapter 4, a performance evaluation is presented as well as some considerations about the system implementation. At last, conclusions, and future work are presented in Chapter 5.

Chapter 2

Background and Related Work

In this chapter, the fundamentals of vehicular ad hoc networks (VANETs) are explained, topics about delay tolerant networks (DTNs) are discussed and state-of-the-art research on data dissemination and file sharing in VANETs is also presented. Firstly, in Section 2.1, a general VANET architecture is reviewed, focusing on dedicated short range communication (DSRC) technology that empowers communication in these networks. Section 2.2 presents a general introduction to delay tolerant networking. In Section 2.3, several data dissemination schemes for VANETs are presented together with an in-depth look at cooperative content download and popular content distribution architectures.

2.1 Vehicular Ad Hoc Networks

Vehicular ad hoc networks (VANETs) comprise a very particular case of mobile ad hoc networks (MANETs) where the mobile nodes are vehicles. Similarly to MANETs, a VANET is a system of self-organizing and self-configuring nodes that act as a host and a router extending the one-hop coverage area of a single wireless node. These nodes can be stationary or mobile, leading to network topology changes.

2.1.1 Architecture

VANETs are comprised of a set of vehicles equipped with on board units (OBUs), enabling them to communicate wirelessly with each other in vehicle to vehicle (V2V) interactions. VANETs can also have road side units (RSUs) that are placed at fixed positions along roads and highways that allow for some structure in the network, enhancing network performance and offering more stable connections for data transmission. OBUs and RSUs can be seen as part of the ad hoc domain. Although not strictly necessary to create a VANET, an infrastructure domain often complements the ad hoc domain, expanding greatly the versatility of the network and providing Internet access. RSUs serve as a gateway to the infrastructure network, enabling vehicle to infrastructure (V2I) communication, illustrated in Figure 2.1. Having RSUs with a reliable and high bandwidth backhaul connection can yield tremendous benefits in the performance of the network. These benefits

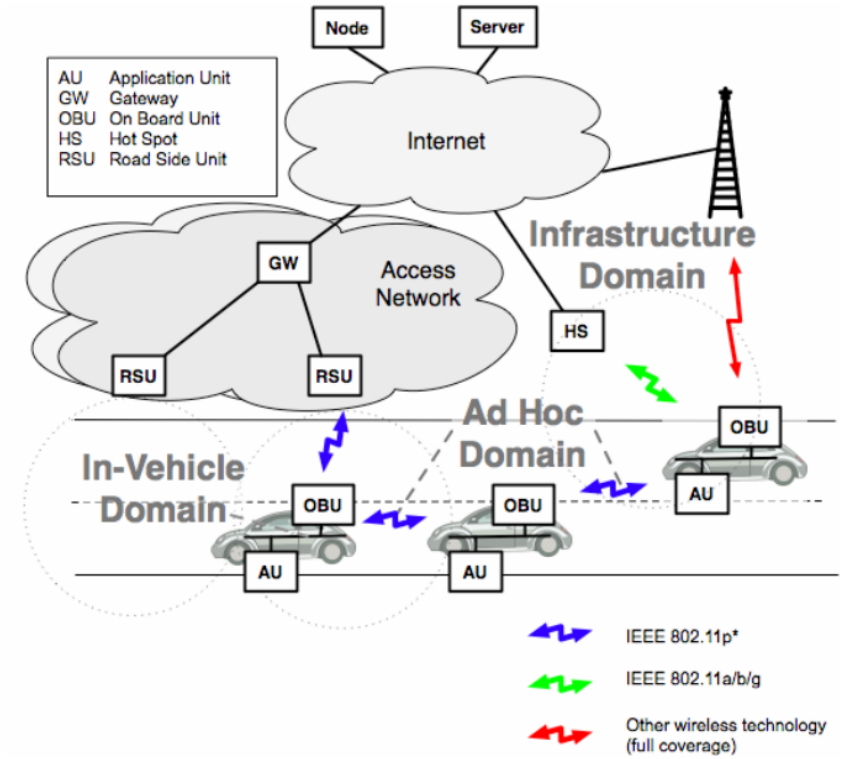


Figure 2.1: Reference architecture (taken from [6])

usually overcome the high investment needed for their deployment. However, these units are often scarce and their placement needs to be well planned; many studies have been conducted on optimizing the placement of these units [3, 4, 5].

2.1.2 DSRC/WAVE Standards

OBUs and RSUs have limited wireless range and, as such, the high mobility of vehicles leads to a very small time frame for interactions between them to occur [7]. Tradicional wireless communication protocols are not able to cope well with the dynamic nature of VANETs nor to comply with the strict requirements these networks demand. To alleviate this problem, dedicated short range communication (DSRC) and wireless access in vehicular environment (WAVE) were designed as a set of standards to improve the connection establishment and maintenance in VANETs. These improvements are crucial to vehicular safety applications that cannot tolerate long delays in the connection establishment that might inhibit successful communication between vehicles. Also, for non-safety related applications, these standards allow for several services to be implemented that can enhance the effectiveness of intelligent transportation systems.

The WAVE protocol stack and architecture is represented in Figure 2.2 and its main constituents are summarized as follows.

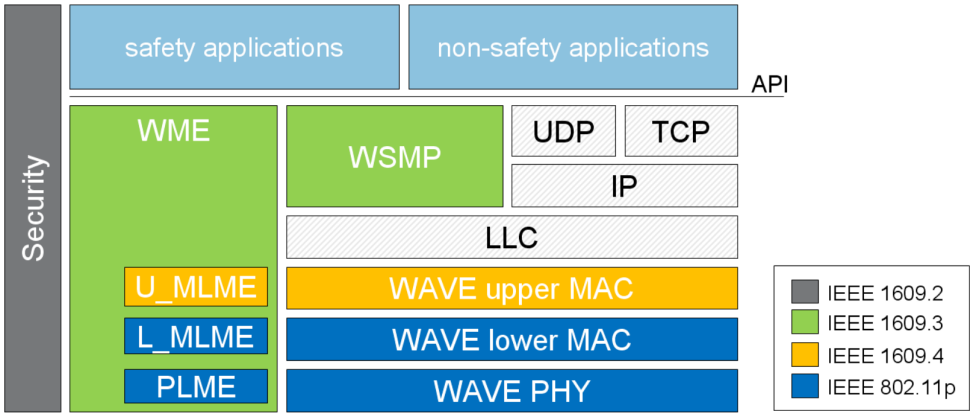


Figure 2.2: WAVE protocol stack (taken from [8])

- **IEEE 1609.0** – "Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture.";
- **IEEE 1609.1** – "Standard for Wireless Access in Vehicular Environments (WAVE) - Resource Manager.";
- **IEEE 1609.2** – "Standard for Wireless Access in Vehicular Environments (WAVE) - Security Services for Applications and Management Messages.";
- **IEEE 1609.3** – "Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Services.";
- **IEEE 1609.4** – "Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-Channel Operations.";
- **IEEE 802.11p** – "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications – Amendment 6: Wireless Access in Vehicular Environments";

At the PHY layer DSRC utilizes the 802.11p amendment to the IEEE 802.11 standard which only adopts OFDM on 10 MHz channels operating at the 5.9 GHz frequency band. In Europe, the band 5855-5925 MHz has been designated for the implementation of intelligent transportation systems (ITS). The sub-band 5855-5875 MHz has been reserved for ITS non-safety applications, the sub-band 5875-5905 MHz for safety-related ITS applications and the sub-band 5905-5925 MHz as an extension of ITS spectrum. Other frequency bands have been defined for use with DSRC around the world, of particular interest are the ones shown on Table 2.1.

At the MAC layer, WAVE uses IEEE 802.11p as well as a layer extension defined by IEEE 1609.4. This extension provides multi-channel operation making use of the concept of frequency/time division multiple access (FDMA/TDMA) to support a control channel (CCH) and multiple service channels (SCH). There are two main modes of operation as shown in Figure 2.3: (a) continuous mode and (b) alternating mode. The control channel (CCH) frequency and time slot is used to

Table 2.1: Spectrum Allocation for WAVE/DSRC Applications (based on [9])

Coutry/Region	Frequency Bands (MHz)	Reference Documents
ITU-R (ISM band)	5725-5875	Article 5 of Radio Regulations
Europe	5795-5815, 5855-5925	ETS 202-663, ETSI EN 302-571, ETSI EN 301-893
North America	902-928, 5850-5925	FCC 47 CFR
Japan	715-725, 5770-5850	MIC EO Article 49

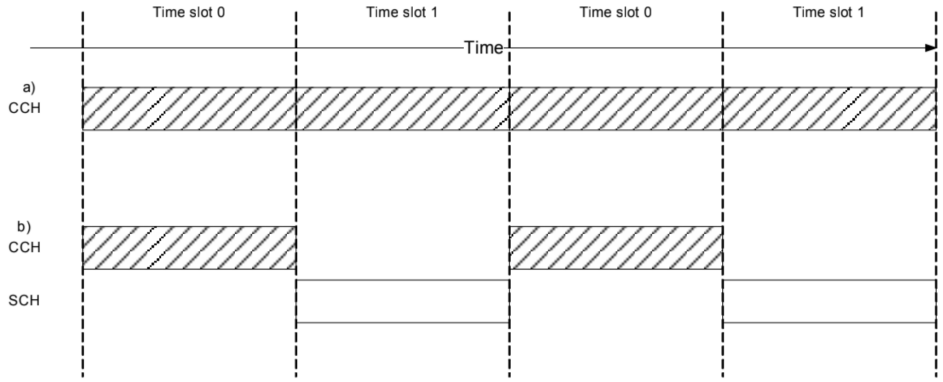


Figure 2.3: Channel access examples: (a) continuous and (b) alternating

transmit either (a) short messages, primarily for safety applications, as defined by the WAVE short message protocol (WSMP) or (b) WAVE service advertisement (WSA) messages used to announce the services available on other SCH frequency channels [9]. Moreover, the CCH only allows for messages to be sent in broadcast, which implies that a SCH must be used whenever unicast messages need to be exchanged.

WAVE short message protocol (WSMP) is a transport and network layer protocol which is described in IEEE 1609.3. This provides the application layer protocol IEEE 1609.1 with the ability to determine physical layer characteristics such as channel number and output power, as well as data rate and receiver MAC addresses. The 1609.1 protocol is a resource manager that multiplexes communication between one sender and multiple receivers. The IEEE 1609.2 protocol adds security which is very important in this kind of networks [10].

2.2 Delay Tolerant Networks

Due to some fundamental assumptions built into the Internet architecture, such as relatively static topologies and the need of a constant end-to-end path between two nodes, traditional Internet protocols do not work well for disruptive scenarios. Common routing protocols designed for MANETs, such as OLSR, AODV and DSR, permit mobile nodes to obtain routes quickly for new destinations and respond to changes in network topology in a timely manner. Yet, they assume

an end-to-end path between any pair of nodes, small round-trip times (RTTs) and low drop probability. In very dynamic MANETs, high node mobility leads to a constant change of the network topology that can violate all the assumptions mentioned. Delay Tolerant Networks (DTNs) are built with these issues in mind.

DTNs are designed to carry out communication in adverse conditions with sparse and intermittent connectivity, large and variable delay, high error rates, and even no end-to-end path between nodes. With the absence of a complete path between source and destination nodes, the data can start to be transmitted, getting stored in a node when no path is available and forwarded when a new link is established. This concept is called store-and-forward and it implies that the DTN nodes must have some type of persistent storage. Also, DTN nodes transfer the responsibility for the data, i.e. there are no end-to-end acknowledgments, only link acknowledgments.

Cerf, et al. [11] proposed the use of a message-oriented overlay called bundle layer below the application layer to manage the data to be sent. This layer transforms application data into units with a defined format called bundles which are forwarded by the nodes. Bundles may be split up into multiple smaller bundles during transmission and reassembled anywhere in the network.

Scott and Burleigh [12] proposed the use of a convergence layer that abstracts the characteristics of lower layers to the bundle protocol making possible a reliable transfer of bundles between two DTN nodes independently of the set of lower layers in use. This convergence layer is also in charge of sending and receiving bundles on behalf of the bundle protocol that generally deals only with the forwarding phase and does not provide routing details.

2.3 Vehicular DTNs

In VANETs vehicles have a high mobility and are distributed over a wide area, therefore is common that there is no end-to-end path between source and destination. Hence, a vehicular delay tolerant network (VDTN) is often the solution applied.

2.3.1 Routing

As stated in section 2.2, generally, the bundle protocol deals only with the forwarding phase, it does not provide details of routes for the data packets between the nodes [13]. Routing in VDTNs is a very important topic and many studies have been carried trying to provide adequate solutions.

Tornell, et al. [14] proposed a protocol taxonomy described bellow and summarized in Figure 2.4. The first hierarchy categorization is due to the objective of the protocol, it can be a) Unicast – messages have a specific destination or b) Dissemination – messages should reach all nodes in the network. In the second categorization level the protocols are grouped according to the amount of control information required. In the dissemination category there are epidemic protocols and geo-connectivity protocols, which estimate node connectivity based on geographic information. Unicast protocols that do not require knowledge about the vehicle or environment status are categorized as Zero Knowledge, otherwise are categorized as Utility Based. The protocols in the latter category choose the routes based on an estimation of how a transmission improves

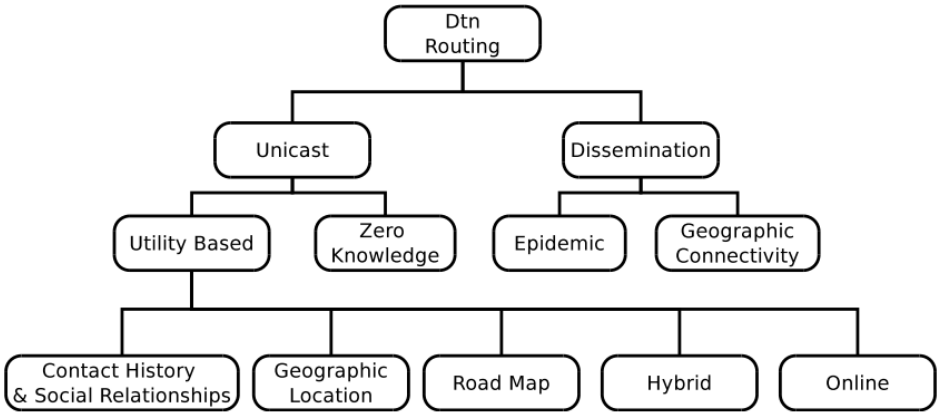


Figure 2.4: DTN protocol taxonomy (taken from [14])

the probability of reaching the destination, this metric is called benefit of transmission and can be calculated based on different knowledge: i) contact history and social relationships, ii) geographic location, iii) road map, iv) hybrid protocols and v) online protocols.

2.3.2 Data Dissemination

Many studies have presented strategies that disseminate messages across a vehicular network. Message dissemination schemes focus on selecting a data path that has the best chance to spread the message over a large number of nodes inside the network in an efficient way.

A Flooding scheme [15, 16] where each node rebroadcasts every message received certainly accomplishes the dissemination requirements of reaching as many nodes as possible. However, besides the known issues of broadcast storm [17] and waste of resources in infinite rebroadcast loops, it is also limited by the connectivity of the network because nodes will only propagate messages as long as the network is connected.

Epidemic routing [18, 19], shares a message every time a contact occurs, if the encountered peer does not have it yet. This requires contents to be stored which, depending on the policy used, may use too much storage. Moreover, it needs a trigger to stop the dissemination and it also requires a negotiation phase which introduces some overhead and delay.

Geographic Protocols [20, 21, 22, 23, 24, 25] are protocols who base their routing in positional information. DSCF [20], FFRDV [21] and DV-CAST [24] are examples of protocols only suited for highway scenarios due to their directionality where they make decisions based on whether the nodes are moving towards or away from the message source.

Road Oriented Dissemination (ROD) [22] uses the same principle as Distance Defer Transfer Protocol (DDT) [26] to optimize bandwidth usage. DDT selects only one vehicle per transmission to rebroadcast the message, ROD adds a store-and-forward mechanism when no vehicle is able to disseminate the messages further, thus rebroadcasts periodically until another node has also received and broadcasted the message. UV-CAST [23] is similar to ROD but, instead of overhearing

messages from its neighbors to choose the carrier nodes, it uses geographic position to determine whether they are located on the boundary of the source node's connected region. SERVUS [25] joins groups of connected nodes in clusters based on their geographic position and if a node is the last node of the cluster rebroadcasts previous messages when it contacts a new node from outside of the cluster.

These protocols address the dissemination of data inside a network. Some focus on reducing the dissemination time, others try to optimize bandwidth and reduce congestion. However, they do not specify efficient ways of distributing large contents that originate from outside the network.

2.3.3 Peer-to-Peer and Colaborative Download

In VANETs, vehicles high speed and connections limited bandwidth means that often OBUs fail to download the entirety of the content when passing through an RSU. Given the long inter-RSU distances, vehicles then move into areas of transmission outage that delay the download of the content. Cooperative networking was first introduced as a way to cope with the web flash crowd problem, improving network performance [27]. However, it can also be applied in VANETs to reduce the download time of a file by allowing a node to continue fetching the required content from neighbor nodes when in areas without access to RSUs.

SPAWN [28], a pull-based protocol, is proposed as a BitTorrent-like strategy where the content is divided into smaller blocks of equal size, called chunks or pieces. The protocol starts with a car arriving in the range of a gateway (RSU); it initiates the download receiving as many chunks of data as possible during the contact duration; after getting out of range it starts to gossip with its neighbors about the content it possesses and exchanges pieces of the file; as long as its neighbors have chunks of the file, it can keep downloading the file as opposed to waiting for the next gateway to resume the download. To advertise the chunk list that it possesses, the OBU periodically sends a gossip message containing the TorrentID that identifies the file, the list of pieces, a timestamp of when it originated and a list of nodes that processed it along the route. The periodicity of this advertisement has the potential of generating a large number of gossip messages increasing network congestion. The authors also showed that a rarest-closest first approach to the chunk selection strategy, where each node first determines the rarest file piece it needs and then looks for the closest node that has it, performs much better than the default rarest-first strategy used in BitTorrent, and that the locality awareness of this approach improves the scalability and performance of the P2P network. Simulation experiments in [29] further support that the best chunk selection strategy combines closeness and rarity.

Studies [30] and [31] address the V2V phase of collaborative PCD problem. Wang et al. [30] propose a solution based on coalition formation of game theory. In this approach, neighbor nodes join and collaborate to maximize a utility function that combines content requests, peer locations, channel capacities and potential interferences. If the network scale surpasses a given threshold, it is divided into several sub-networks. For each sub-network, the nodes group into several coalitions, and the nodes in the coalition with the highest service rate broadcast chunks based on a greedy strategy. In [31], the authors propose the grouping of nodes into clusters (cells) based on their

position. To achieve this, roads are divided into continuous homogeneous cells of fixed size and position. This allows to treat the cells as static nodes which simplifies the modeling of the network and eases the decision process of where to disseminate the content to. Cell density is defined as the average number of OBUs present on that cell. The use of principles such as high cell density first and high downloading rate first accumulate chunks to a small number of core cells first, and afterwards distribute them to low-degree cells within a few hops. Simulation results show that this approach performs better that the coalition formation approach presented in [30], especially with large content sizes.

Other studies do not focus heavily on the content dissemination across many nodes, but nonetheless use cooperation strategies the overcome the limited range of RSUs. These deal with the transmission outage problem with local store-carry-forward transmission mechanisms that select some vehicle to act as a relay. The authors in [32] proposed an opportunistic relay protocol, where the RSUs broadcast the data packets such that all vehicles in its range can hear it. This data broadcast is done at an optimal data rate that strikes a balance of throughput and distance, in order to ensure that only the best suited vehicles successfully receive the data, and thus become relay candidates. Each candidate contends to relay the frame to the destination, and through an analytical model the expected throughput to the destination is computed. The vehicle with higher expected throughput, usually the one closer to the destination, is selected as the relay vehicle. Unlike [32], in [33] the authors focus on a bidirectional highway scenario. They propose a store-carry-forward scheme that utilises inter-RSU cooperation to select a second relay from the vehicles moving in the reverse direction. Having two consecutive relays, one in each direction, greatly reduces the transmission outage time.

In [34] the authors propose an approach with a query management server that uses vehicle location data to forward the pending request to RSUs in the area where the downloader vehicle is traveling. Based on contact information and download rates from the server, RSUs predict future V2V and V2I contacts and make locally-optimal decisions on where to forward the data. The data can be delivered directly or by a relay vehicle; its assumed that all vehicles are available for traffic relay whenever they are not receiving data from an RSU.

Many network coding based approaches have also been proposed like VANETCODE [35], CodeTorrent [36], CodeOn [37], CodeCast [38], and the ones presented in [39, 40]. These proved to have good overall performance; however, these solutions have more complex implementations and are not ideal for resource constrained devices since communication overhead, computational overhead and disk I/O overhead can become problematic [41]. Also, often the performance of network coding approaches are not dissimilar from non-coding approaches [31].

Chapter 3

Proposed System Architecture

This chapter describes the proposed system architecture and presents the design decisions, complemented with the rationale behind them. Firstly, a characterization of the problems addressed by this work is presented in Section 3.1. An overview of the solutions to these problems is presented in Section 3.2. Section 3.3 provides an in-depth look at all the constituents of this architecture and the fundamentals that motivated the design decisions. To wrap up this Chapter, implementation details are described in Section 3.4.

3.1 Problem Characterization

In a general form, the problems addressed by this work can be divided into three main parts: (1) content discovery, (2) peer discovery and (3) peer and chunk selection. These three phases have very distinct goals and requirements and thus are handled in different ways.

This work is focused on disseminating over-the-air (OTA) updates, which are quietly made available in the cloud. Therefore, a node is not aware of its existence, and thus there must be a content discovery mechanism that informs OBUs and RSUs of this content. Veniam already has a solution for content discovery which involves polling the cloud periodically. However, the current cloud infrastructure does not implement the features needed by this work, namely the creation of chunks and metadata. Therefore, a new cloud infrastructure needs to be implemented and, consequently, a content discovery mechanism independent from the one in production also needs to be created. There is also security concerns about content access and data integrity that need to be addressed.

After discovering the required content, a node needs to learn which peers can provide him that content. Therefore, a peer discovery mechanism that provides the information needed for the selection mechanism to decide who will be the content provider, needs to be designed. The peer discovery mechanism must get an up-to-date information, rapidly and without generating much network traffic, allowing for the scalability of the network without running into congestion problems. A peer and chunk selection algorithm also needs to be developed to facilitate the chunk exchanges and avoid connections that might decrease the dissemination performance.

3.2 Solution Overview

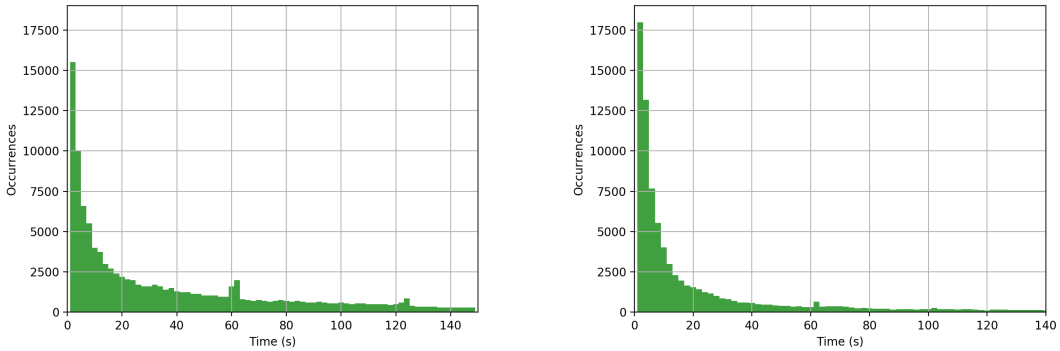
Similarly to the already implemented content discovery mechanism, this work also periodically polls the cloud infrastructure to verify if an update is available. The details of this interaction are explored in Subsection 3.4.2.

In order to overcome the problems of diversified contact duration inherent to VANETs, the approach followed in this work bears some resemblance to P2P file sharing networks, such as the well known BitTorrent. Specifically, it applies the concept of dividing the content into several blocks of equal size, called chunks. The chunk size is an important parameter that can have a substantial influence in the performance of the dissemination. Small sized chunks imply that a file will be split into many chunks which causes an increased overhead for two reasons: (1) the control messages become larger as they need to represent numerous chunks and (2) the overhead inherent to the TCP protocol is exacerbated as the number of connections needed to transmit the entire file increases. On the other hand, large chunks take longer to be transmitted which increases the probability of a disconnection happening during the exchange, causing data to be discarded. The incomplete chunks are discarded so that chunks can be treated as indivisible units identifiable by an ID number.

The content segmentation process happens prior to any data exchange and, as such, it should be controlled by the entity who owns the original content, which in this work is the cloud infrastructure. Together with the process of fragmenting the designated data into chunks, the cloud also generates the meta information of the content to distribute. This is called metadata and it is the first thing a node receives when it initiates the process of downloading the content. Also, the distributed nature of vehicular networks raises some security concerns especially about data integrity. These are addressed by validating each dissemination attempt in the cloud and using a hashing function to perform a data integrity check.

The peer discovery is achieved by periodically broadcasting a probe to all neighbors within one hop distance asking for a given content. The neighbors respond to theses probes, specifying the set of chunks possessed for that specific content. A node ceases to send these probes after it has completed the download, but it continues to answer to neighbors probes until the content's time to live (TTL) has expired.

In regards to the peer and chunk selection, two approaches are followed: (a) chunks are chosen sequentially and the first peer to respond is selected, and (b) peer and chunk are chosen randomly. Both approaches are complemented with a backoff algorithm that uses intelligence from the last chunk exchanges to avoid peers that showed to be problematic. In this work, RSUs and OBUs are treated equally and all nodes take part in the P2P overlay. However, there is a configurable parameter that allows a node to get the chunks directly from the cloud. Given the fast and reliable backhaul connection of RSUs, it makes sense to have this feature activated for them.



(a) V2I connection duration

(b) Interval between consecutive V2I connections

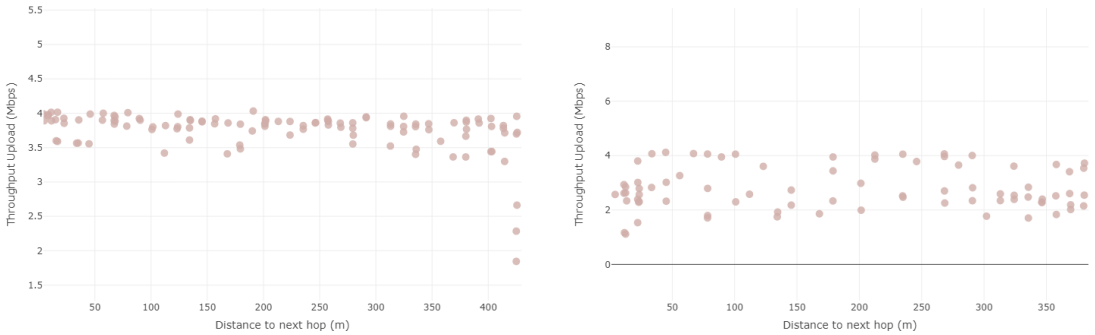
Figure 3.1: Frequency histogram of V2I connections.

3.3 Protocol Achitecture and Design

3.3.1 Chunk Size

Defining a contact as the scenario where two nodes are within range and are able to communicate with each other, then the two most important metrics to take into account when calculating an adequate chunk size are (a) the duration of the contact between nodes and (b) the throughput of the link. Given the availability of a real VANET, all decisions were based in real world data from previously conducted tests rather than theoretical values. Given the current features of the platform, no data was available regarding V2V contact duration. However, one week of data from 366 vehicles in downtown Porto, allowed us to conclude that the duration of OBU-RSU established connections had a mean of 57 s and a median of 20 s, while the interval between those connections showed a mean of 65 s and a median of 9 s. The frequency histogram of the duration of V2I DSRC connections and interval between consecutive connections can be seen in Figure 3.1. As can be seen from the distribution in the histograms, it is very common to have frequent yet short duration contacts, this further motivates the need of a protocol like the one proposed in this work to facilitate the content distribution. Note that there are a few peaks around 60 s and 120 s, this is an artifact of the method used to obtain this data that is based on the established connections instead of the visibility of the two nodes. Also, this was due to an unwanted behavior on Veniam's connection manager, that was later fixed. It caused a connection to be terminated if no response to the keep-alive packet, which has a periodicity of 60 s, was received from the mobility controller. All in all, this does not influence the results much.

Additionally, data on the throughput of a DSRC connection was also available from previous tests. These were conducted with a DSRC channel rate of 6 Mbit/s and without other communications occupying the channel. The throughput was measured using `iperf` in two different scenarios. The first scenario represents a V2I communication with a fixed RSU and the vehicle moving at 50 km/h, while the second scenario represents a V2V communication where one vehicle is stopped and the other one is moving at 50 km/h. In both scenarios, the moving vehicle



(a) V2I at 50 km/h (b) V2V at 50 km/h

Figure 3.2: Throughput vs distance.

started and stopped side by side with the other vehicle or RSU. It performed 3 trips moving away from the other vehicle or RSU, and 2 trips moving towards it. The V2I scenario yielded a median throughput of 3.84 Mbit/s and the V2V scenario showed a median of 3.26 Mbit/s. The distribution of the results obtained for the two scenarios is shown as a function of distance in Figure 3.2. Note that these tests were conducted in a controlled environment. In a real network, others nodes would generate data that occupies the DSRC channels, which might cause congestion, leading to a reduction of throughput.

With T being the contact duration, R the throughput of the link and n the number of chunks transmitted per contact, then the chunk size L_c is given by:

$$L_c = \frac{RT}{n} \tag{3.1}$$

Based on the results presented, assuming a contact duration of $T = 20\,\text{s}$, let the throughput be the most conservative value of $R = 3.26\,\text{Mbit/s}$ and define that a minimum of $n = 8$ chunks must be successfully transmitted per contact. Solving Equation (3.1), 8 Mbit or 1 MB can be inferred as an adequate value for the chunk size.

3.3.2 Security

There are three possible approaches to content distribution: (1) centralized, (2) fully distributed and (3) hybrid. The centralized approach involves a central unit that provides the content to all other nodes. In the fully distributed approach, all nodes have a similar role where they can request or share some content without the participation of any central control unit. In order to avoid a possible congestion of unwanted content requests across the network, a hybrid dissemination scheme was chosen. This scheme relies on the cloud infrastructure to validate all requests before any messages get disseminated. In this manner, when a node wants to access a designated content it must communicate with the cloud, validating that the required file exists and that it has permission to access it, and only then can it continue to download the file.

Given the volatile nature of vehicular networks communication links, it was mandatory to find a way to ensure data integrity. TCP was selected as the transport layer protocol over which the chunk exchanges are performed, due to its reliability features, error detection, flow control and congestion control. These mechanisms are very important when transmitting data on crowded areas, where the DSRC communication channels are prone to congestion. TCP has a 16-bit ones-complement checksum mechanism that will catch any burst error of 15 bits or less, and all 16-bit burst errors except for those which replace one 1's complement zero with another (i.e., 16 adjacent 1 bits replaced by 16 zero bits, or vice-versa) [42]. However, when moving large amounts of data over adverse network conditions, this is not enough. Even cyclic redundancy check (CRC) mechanisms are unreliable to ensure data integrity [42, 43, 44]. The adopted solution was to use a cryptographic hash function to calculate a hash of each individual chunk and another one of the whole file. When a chunk exchange is finished, its hash can be computed and compared with the expected one, as specified in the metadata. In this manner, it can be immediately verified if the received data corresponds to the expected one and decide whether it should be kept or discarded. Several hashing functions were taken into account, SHA-256 is a very popular one, and although some weaknesses are already known [45], these are related to possible collisions that affect signatures but are irrelevant for data integrity checks; thus, SHA-256 was the hashing algorithm chosen. In addition to the verification of each individual chunk's hash on its arrival, the subsequent comparison of the whole file's hash adds an extra layer of assurance that is even able to identify possible storage errors or malfunctions. In case the calculated chunk hash does not match the expected one, the problematic chunk is discarded and later requested again. On the other hand, if a file hash does not match then the individual chunk hashes are recalculated and the chunks whose hash do not match are requested again. If this situation is recurrent, it might be caused by an error in the metadata received or a defect in the node's storage. To deal with this, the file is blacklisted and it will not be requested again, freeing the DSRC channel and avoiding unnecessary congestion on the network.

3.3.3 Cloud and Content Discovery

The cloud infrastructure has an important role in this work. It segments the content into chunks, validates the download requests, generates the content's metadata and provides an API that allows nodes to check for new content, retrieve chunks and metadata.

The first phase in the dissemination process is to access the cloud infrastructure in order to validate the request and to retrieve the metadata. This interaction happens only once per content and the amount of data involved in it is quite small, just a few kilobytes. Also, it is important that this first stage is completed rapidly so that the second stage starts as early as possible. Given these requirements, it was decided that this access to the cloud and consequent download of the metadata should be accomplished using the communication technology available at the moment, even if it involves using the expensive cellular connection.

Nodes have a configurable parameter that allows them to download chunks directly from the cloud. Veniam's network has the support of an infrastructure of RSUs with cheap, fast and reliable

backhaul connection; thus, it makes sense to allow these units to access the cloud to download the content.

When a node requests some content it knows the content's name or ID but there is some additional information inherent to the content that is required to successfully download it using the dissemination protocol. This information comprises the following fields: file ID, file name, file size, file hash (Section 3.3.2), chunk size and an array of all chunk hashes. There are many serialization formats that could be explored to transmit this data but, realistically, in this particular application, this is not specially important and thus it was decided that the meta data would be serialized into JSON format, as it is standard at Veniam.

3.3.4 Peer Discovery

After getting the required metadata from the cloud a node may start the second stage, which encompasses the exchange of control messages to assess the chunk availability in adjacent nodes. There are two approaches that can be implemented: (1) gossiping and (2) probing.

In a gossiping strategy like the one proposed in SPAWN [28], a node announces its own content to its neighbors, which then decide whether to download it. On the contrary, in a probing approach, the interested node is the one that sends messages to its neighbors, which then respond with a list of chunks they possess. In gossiping, a node keeps generating network traffic after the file has finished downloading, which at scale can be a significant problem, and also requires extra coordination with the cloud infrastructure to define when the nodes should stop gossiping. On the contrary, in the availability probing mechanism, nodes stop sending the control messages after finishing the download, which reduces congestion as the dissemination proceeds. For this reason, the peer discovery strategy adopted in this work is probing.

In order to keep an always up-to-date record of the content in its ever-changing surroundings, this message exchange is repeated periodically until the node has completed the download of the content. This action continues to be executed while a chunk is being downloaded, allowing for an immediate decision to be taken if the download fails and another chunk-node combination needs to be selected. The probing periodicity is configurable and a value of 500 ms is reasonably fast to ensure that a node is always aware of its dynamically changing surroundings without becoming unbearably demanding in terms of its internal resource usage.

It was evaluated whether these probes should take advantage of DSRC provider services or if it should use UDP signaling. Using the DSRC provider services, channel congestion is smaller because WAVE Service Advertisements (WSAs) are bundled into a single frame, which avoids extra transmissions and provides less overhead, due to it being L2 frames. If the network is operating in alternate mode, then the WSA frames are sent in the control channel (CCH), avoiding the traffic from other applications in the system. However, the testbed where this work was tested was operating in continuous mode which invalidates this benefit. Also, using UDP to perform the availability probing makes this protocol agnostic to the communication technology being used, which is an important benefit in platforms that use multiple interfaces, e.g. Veniam platform also uses Wi-Fi for inter-node communication. Moreover, in this scenario, the mentioned DSRC

Table 3.1: Datagram

Offset octet	0	1	2	3	4	5	6	7	8	9 ...
	Type	Node ID				Number of Blocks				Data Blocks

Table 3.2: Data block for chunk list

Offset octet	0	1	2	3	4	5	6	7
	File ID				Chunk ID			

provider service benefits only have a practical effect on high congestion situations, which greatly affects the chunk transfers either way. With these effects in mind, it was decided to use UDP for its versatility.

Multicast could be used to restrict the reachability of the probes, creating groups of nodes that are able to receive these messages. However, the scenario tested is one where there is no differentiation between nodes. Given the need to query all neighbor nodes, these availability probes are broadcast over UDP so that all nodes within range receive the message. Upon receiving one of these probes, a node must respond specifying the files it holds or, if the requested content does not match any of its possessions, ignore it. These responses are sent in unicast to the requesting peer in order to avoid that other nodes process unnecessary messages that may not be of their interest.

While an OBU ignores availability probes of contents it does no possess, RSUs start to download the requested content to become a seeder for that file. This network configuration applies a simplified strategy of RSU caching, where data is stored on the edge of the network and can then be disseminated to the rest of the nodes. This is a reactive approach where after every request a RSU keeps the content in storage until the file's TTL has expired. If a request for another content is received and the storage is full, then that file is not cached. This caching policy is far from ideal and is very different from other caching strategies, like the ones present in [46]. However, in the specific scenario of this work, the dissemination is controlled by the cloud infrastructure and it is not expected to have a large number of concurrent content dissemination happening. Therefore, it is an adequate strategy for keeping the content on the network, facilitating the download for other nodes.

In order to support multiple files being concurrently disseminated, the availability probe is a variable length datagram. As depicted in Table 3.1, the first byte defines the type of the message. Currently, there are only 2 possible values for the type field: 0 for when a node is querying its neighbors for their content availability, and 1 for when a node is replying to a received probe. The next 4 bytes contain a unique ID that identifies the node who sent the message. Bytes 5 to 8 represent the number of data blocks described by the remaining bytes starting on byte 9, which are represented on Table 3.2, Table 3.3 and Table 3.4.

The most straightforward approach to represent the chunks needed is to list all the chunk IDs. However, this has an unbearable maximum datagram size L_{max}, given by Equation 3.2, where N is the number of files and n_i is the number of chunks that make up file i. 9 is the of the datagram

Table 3.3: Data block for ranges approach

Offset octet	0	1	2	3	4	5	6	7	8	9	10	11
	File ID				Chunk ID Start				Chunk ID End			

Table 3.4: Data block for bitmap approach

Offset octet	0	1	2	3	4	5	6	7	8 ...
	File ID				Bitmap size				Bitmap

header size, as can be seen in Table 3.1, and 8 is the size of the block represented by Table 3.2.

$$L = 9 + \sum_{i=1}^{N} 8n_i \tag{3.2}$$

To compress the data, two different approaches to express the data blocks were explored, with different advantages and disadvantages. In the first approach, the nodes announce the required chunks in the form of a range of chunk IDs which is defined by a start and end as shown in Table 3.3. The second method uses a bitmap where each bit represents one chunk and can be either 0 if the chunk is not required/available or 1 if the chunk is required/available. As shown in Table 3.4, this second option needs to be complemented with 4 bytes that represent the bitmap size due to the variable number of chunks per file and consequently a variable bitmap length.

The first method has the unique characteristic of having a variable length message that is affected by the state of download; in particular it varies based on the ability to group the missing chunks into ranges. Depending on the arrangement of the chunks it might be possible to group all chunks of a certain file into a single range, thus minimizing the datagram size, or it might be fragmented in such a way that it requires many different ranges to express the state. By choosing to download chunks in a sequential manner it is always assured that the message will have a constant and minimal size, independent of the number of chunks. This approach yields a datagram length L than can be expressed as:

$$L = 9 + 12N \tag{3.3}$$

On the other hand, if the chunk decision is not sequential, this range-based method is no longer ideal. In this situation, this approach needs to represent up to $\frac{n_i}{2}$ ranges per file. Therefore, the datagram has a variable length bounded by an upper limit L_{max}, given by Equation (3.4):

$$L_{max} = 9 + \sum_{i=1}^{N} 6n_i \tag{3.4}$$

In contrast, the bitmap approach has a message length that does not depend on the current state of download, but only on the number of files and the number of chunks per file, as defined by Equation (3.5).

$$L = 9 + 8N + \sum_{i=1}^{N} \left\lceil \frac{n_i}{8} \right\rceil \tag{3.5}$$

If a sequential chunk request is used, the range-based approach can compress the data into a smaller datagram. However, if a non-sequential decision process is chosen, the fixed length of the bitmap-based option is more efficient than the upper-bounded range-based alternative.

3.3.5 Decision Making and Backoff Mechanism

After collecting information about the content availability on adjacent nodes, a node needs to decide which chunk is going to be downloaded and which node will supply it. Two approaches to the chunk selection are considered: (a) request the chunks in sequential order and (b) request the chunks randomly.

The simplest solution is to choose the first node to reply to the probe and request the chunks in sequential order, which pairs well with range-based probes due to their small size. However, this suffers from the well documented last chunk problem in P2P networks [47], where some chunks are very popular and are well distributed across the network while there is a starvation of the least popular chunks. To avoid this problem without increasing the complexity too much, both chunk and node are chosen based on a pseudo-random algorithm.

This algorithm requires additional fine tuning to account for potential problematic behaviors. There are three different scenarios that require specific actions. Assume two nodes, A and B:

- If node A fails to connect to node B because node B is already transmitting some content to another node, then the connection is rejected and node A sets a backoff time of $\frac{t_c}{2}$ during which it does not attempt to reconnect to node B. The expected download time of one chunk t_c, is a parameter that can be calculated based on the chunk size L_c and the expected throughput R, given by equation 3.6.

- If node A fails to establish a connection because it cannot reach node B or if the connection was lost sometime during transmission, then node A ignores previous records of availability from node B. If available, node A tries other nodes, otherwise it waits until a new response to the availability probes is received.

- If node A receives a chunk from node B and its hash does not match the expected one, node A reduces the priority of node B, favoring other nodes. If this situation happens more than N_{tol} times for a certain node, then that node is blacklisted for that file, i.e. it will not be selected for any other chunk from that file. N_{tol} is a configurable parameter and it was set to 3 in the trials conducted.

$$t_c = \frac{L_c}{R} \tag{3.6}$$

3.4 Implementation

3.4.1 Cloud Infrastructure

First, the cloud component of this work was implemented as an HTTP server written in Java. This cloud architecture supports a command line interface (CLI) used to: (1) add new files to the content that can be downloaded, (2) manage the HTTP server and (3) initiate the dissemination trials. It is also through this CLI that the parameters like chunk size, file ID and access permission can be set on a per content basis. In case these parameters are not specified, the default configurations, stored in a JSON file, are applied.

Three different API paths have been created:

- `/download` is where nodes can download the chunks and the metadata.
- `/logs` is the path where nodes upload the data logs after finishing a trial.
- `/check` is the path that specifies a list of active trials at each moment.

When a node wants to download a chunk from the cloud it must perform an HTTP GET request to the `/download` path specifying two parameters: (a) file and (b) chunk, which represent the file ID and chunk ID, respectively. To access the content's metadata, a node performs the same HTTP GET request, but only specifies the file parameter.

3.4.2 Content Discovery Script

The content discovery mechanism was kept separate from the main application to ensure future integration with the rest of the platform. As mentioned in Section 3.1, Veniam already has a solution for content discovery that is planned to be adapted to interact with the application developed in this work.

Therefore, the cloud polling is done via a Python script that, with a periodicity of 60 s, accesses the `/check` API path to verify if any new content has been made available. It also utilizes the inter-process communication (IPC) framework used in the OBUs and RSUs, to communicate to the application specified by this work, that a new file should be downloaded.

3.4.3 Application

The application responsible for executing the protocol specified by this work was implemented as a multi-threaded application written in C. This thread-based architecture is adequate due to the big focus on I/O operations inherent to this type of work. There are six different thread types in this implementation:

- `Request manager thread (main thread)` is responsible for loading all settings from the configuration file, starting all the required threads and interfacing with the IPC framework to handle the content requests. When handling a request from the IPC framework, this thread verifies that downloading the requested file will not exceed the threshold

percentage of the storage that can be occupied. This threshold percentage is defined by a parameter in the configuration file. It is also this thread who is responsible for deleting the file after its TTL.

- `Cloud request thread` is responsible for accessing the cloud to validate the request and to download the meta file required. This thread manages a linked list that the main thread populates with the parsed content requests and it saves the metadata returned from the cloud into the files and chunks tables of the local database, as illustrated in Figure 3.3.

- `Cloud chunk downloader thread`. All nodes are capable of downloading chunks directly from the cloud, yet this type of interaction is dependent on a flag in the configuration file. If this flag is active then this thread downloads the required chunks from the cloud.

- `Availability dispatcher thread` is the central point of this architecture, it serializes the pending contents into one of the datagram formats specified in Section 3.3.4, broadcasts the availability probes, receives and parses the responses from the neighbors and updates the availability and node tables with the information received from the neighbors.

- `Downloader thread` executes the actual decision process based on the information contained in the local database, after which it requests and downloads the selected chunk from the corresponding neighbor. It is also the responsibility of this threads to do the error tracking that feeds the backoff mechanism.

- `Seeder thread` is a TCP server that accepts a connection from another node, reads and validates a request, and transfers the requested chunk.

The number of downloader and seeder threads can have an important impact on the performance of the application, as this parameter controls how many streams of data can happen concurrently on each node, and therefore it is configurable through the configuration file.

3.4.3.1 Database

In order to ensure that the download progress is not compromised by external events, e.g. a reboot, the content and the information relative to its transfer need to be stored in a non-volatile medium. The content itself is stored directly into the OBUs or RSUs storage. The management information like content metadata, download progress and node statistics that feed the backoff mechanism are stored on a local database, implemented in SQLite. This database is comprised of four tables, as show by Figure 3.3 and listed bellow:

- `Files table` holds the metadata of the file. It also holds state information like the `verified` flag that is set to true after the download has finished and the file's hash has been verified. The `path` column holds the path to where the file is stored.

- `Chunks table` holds the metadata of the chunks, a `verified` flag and a foreign key to the files table.

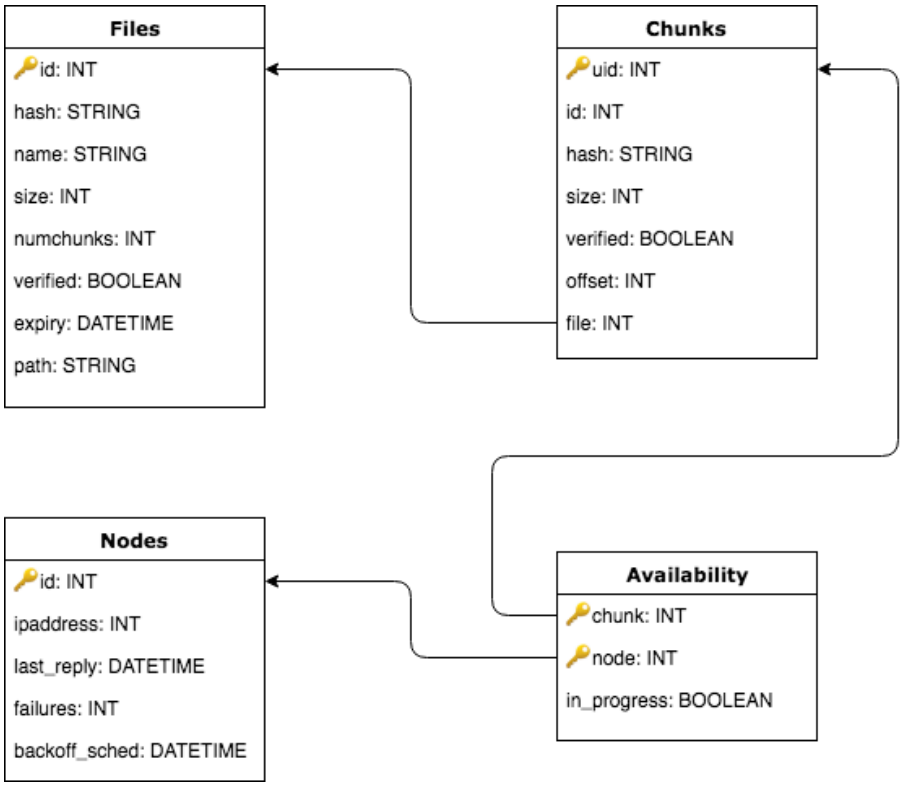


Figure 3.3: Database architecture

- `Nodes table` holds the information of the neighbor nodes that replied to the probes. This data is used to perform the chunk requests and to implement the backoff mechanism.
- `Availability table` holds the node-chunk pairs, represented by two foreign keys to the `Nodes` and `Chunks` tables, as well as a flag that represents if a given chunk is currently being downloaded from the respective node.

3.4.3.2 Configuration

When designing an embedded system's application is mandatory to have the flexibility of changing parameters without patching the application. Therefore, at the start of the application the following parameters are loaded from a JSON file:

- Maximum percentage of free storage space that can be occupied
- Cloud API endpoint
- UDP port
- TCP port
- Number of seeder threads
- Number of downloader threads

- Probing interval
- Download chunks directly from cloud
- N_{tol} (mentioned in Subsection 3.3.5)
- TTL of the availability table records

Chapter 4

Performance Evaluation

4.1 Scenario

The testbed where the experimental tests were conducted comprises 17 public buses and 5 RSUs. Due to restrictions on bus schedules and maintenance stops it was not possible to run the experiments on all 17 buses at the same time and, because of this, the ratio of RSUs ranges from 25% to 35.7% of the total number of nodes. The exact number of vehicles, methods and file sizes for each trial can be consulted on Table 4.1.

To automate the test deployment, all nodes were set to periodically check the cloud for new tests which implies that, after being started through the cloud, all nodes receive the request within 60 s. Moreover, after finishing the download the nodes upload a log file to the cloud for data analysis.

There are other parameters that were left as configurable and should be mentioned. To simplify the evaluation and keep the parameter space under control, every node was limited to only one seeder and one downloader thread, which means that there was only one upload and one download stream concurrently available. Also, the probing period was set as 500 ms and the time to live of the availability table records was set to 2 s. It was chosen to use files of 40 MiB and 100 MiB split into chunks of 1 MB; 40 MiB is a typical size for an OBU update and 100 MiB was chosen simply to see how the protocol behaves with a larger file.

It is also important to emphasize that the bus routes could not be controlled and, as such, the mobility pattern of the vehicles varies in each trial. Moreover, the trials were conducted at different times of the day, but given the duration of the tests, all of them overlap with peak traffic hours.

Table 4.1: Trial settings

Probing + Decision	Range + Sequential									Bitmap + Random			
File Size	40 MiB				100 MiB					40 MiB		100 MiB	
Trial	1	2	3	4	5	6	7	8	9	10	11	12	13
Vehicles	13	13	15	15	14	15	13	14	11	11	13	9	12

4.2 Performance Metrics

A variety of metrics can be analysed to evaluate the performance of a dissemination protocol:

- `Dissemination time` is the most important metric, because it represents the time needed to make the content available for all nodes. It is characterized by the difference between the moment the last node successfully receives the final chunk and the instant when the content was made available in the cloud.

- `Download time` is the time a node spends downloading the content, i.e. the difference between the moment a node receives the last chunk and the instant when it performed the request to download the metadata from the cloud.

- `Number of chunks exchanged per contact` is a metric that allows for a better understanding of the type of contacts endured during the experiments. It also allows us to see how V2I and V2V links differ in practice.

- `V2V chunk exchange percentage` is important to understand how much V2V interactions are contributing to the dissemination. This metric shows the impact of using a cooperative scheme. It is defined as the ratio between the number of chunks exchanged through V2V link and the total number of chunks disseminated.

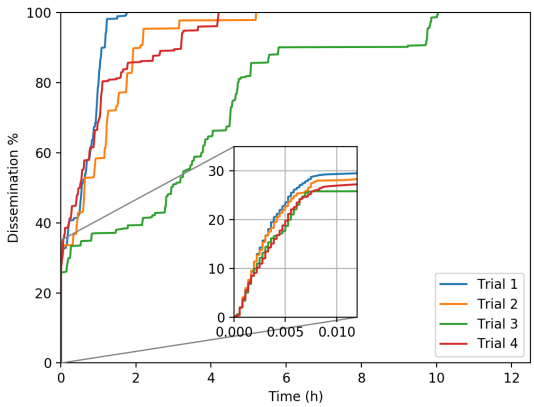
- `Failure ratio` is defined as the ratio between the number of failed and successful chunks exchanges. This is important to analyse the problems faced in the dissemination. Subdividing this metric in the several failure causes eases the optimization of this work.

- `Probing overhead` is defined as the ratio between the total amount of data exchanged by the peer discovery mechanism and the total amount of useful content data. This is crucial to understand the efficiency of the probing mechanism.

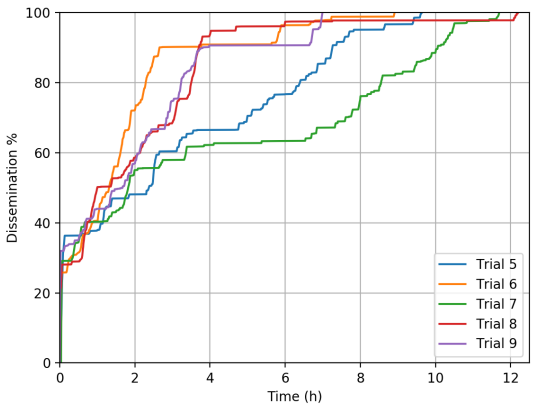
- `Retransmission overhead` is defined as the ratio between the total amount of data discarded due to incomplete transfers and the total amount of useful content data. This metric allows us to infer the impact that the network conditions have in the dissemination process.

4.3 Dissemination Evolution

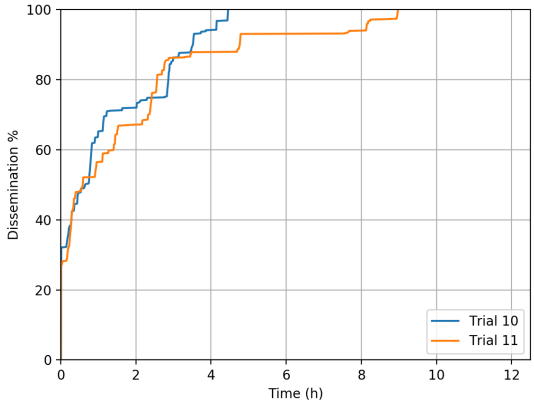
Concerning the dissemination of content across the network, several different behaviors can be identified (see Figure 4.1). Initially, there is a very fast progress due to the fast and reliable backhaul connection of the RSUs that allows them to get the content in a short time, limited only by their pooling interval. This achieves the desired effect of pushing the content to the edge of the network first and disseminating it to the mobile nodes afterwards.



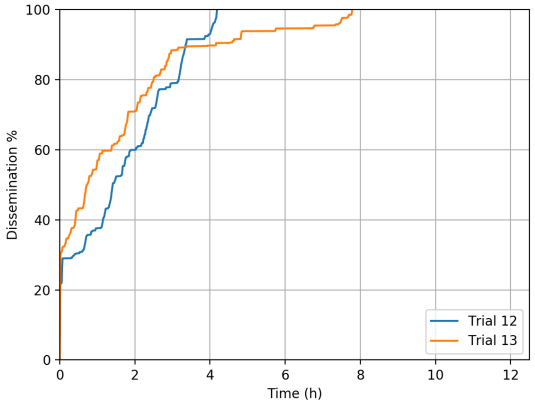
(a) 40 MiB file, range based and sequential



(b) 100 MiB file, range based and sequential



(c) 40 MiB file, bitmap based and random

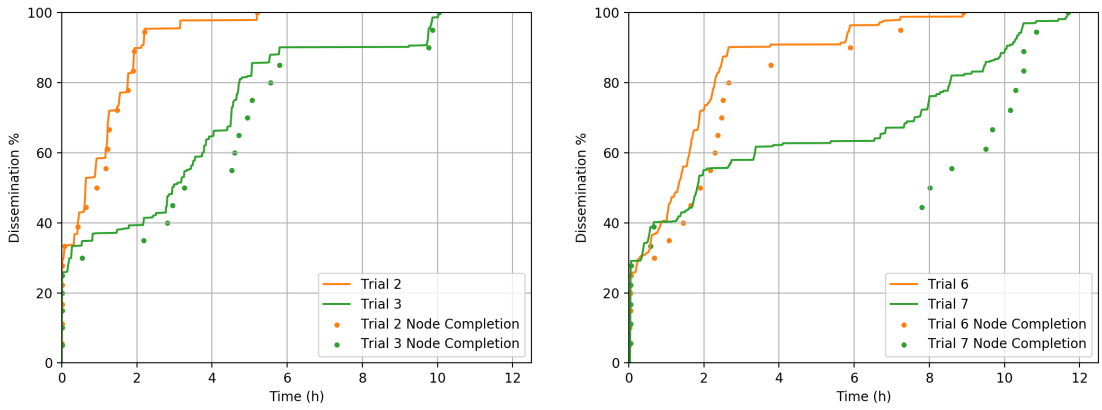


(d) 100 MiB file, bitmap based and random

Figure 4.1: Dissemination evolution with time.

It should also be pointed that, even for equally sized files, there is a large discrepancy in the total dissemination time, e.g. trial 3 took 5.7 times longer than trial 1 to complete the dissemination. Moreover, on most trials, two different phases can be identified: first there is a quick and almost linear dissemination and, after a certain point, there is a slowdown where the dissemination nearly stalls. This might seem counter intuitive at first as one would expect that, with more nodes possessing the content, it would be easier for a requesting node to find a neighbor with the desired chunks, and thus the dissemination would speed up. However, evaluating an animation of the GPS traces of the routes taken by the buses revealed that some vehicles had particular routes that rarely encountered other nodes and thus a tail is present on most trials.

Figure 4.2 shows an overlay of dots that illustrates the moment where a node finishes the download. The vertical axis of this overlaid dots represents the percentage of nodes that have completed the download. It can be clearly seen that almost all nodes have completed the transfer and the tail is caused by a few "lagging" nodes, one in trial 2 and three in trial 3, for example. The vertical distance of the dots to the evolution line in Figure 4.2, represents the number of chunks



(a) 40 MiB file

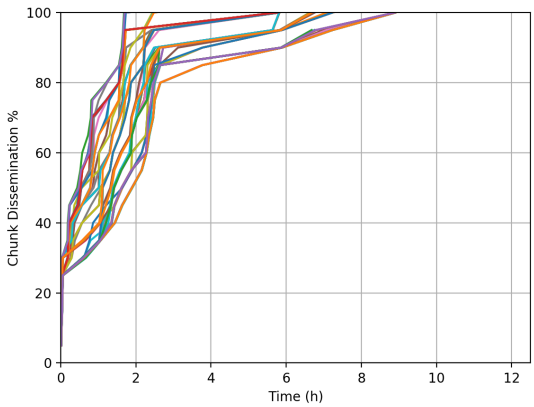
(b) 100 MiB file

Figure 4.2: Dissemination evolution and node completion.

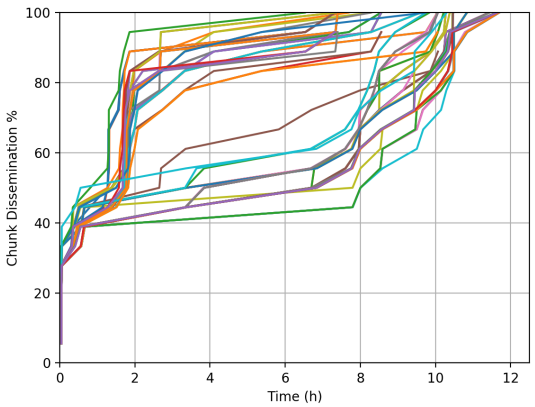
that are present in nodes that have not finished the download of the entire file, i.e. if the dot is far from the line then there are several nodes that possess some chunks but have not downloaded the completed file. In trial 2, the node completion dots follow almost exactly the evolution line, which means that each node was able to finish the download in a few contacts or at least before other nodes where able to exchange any chunks. On the contrary, trial 3 shows that, most of the time, there were several nodes with an incomplete file, the cause for this will be explored further on Section 4.7. As expected, with bigger file sizes this presence of incomplete files is more prominent, as shown by the increased distance between the dots and the line in Figure 4.2b. It is also important to note the difference between trial 6 and 7, where in trial 6 nodes finished the download at a relatively steady pace as opposed to trial 7 where the large time frame between 0.7 h and 7.7 h showed that chunks were being exchanged but the nodes were not completing the download of the file. This delay in finishing the download can be caused by short and sparse contacts that do not allow for many chunks being transmitted per contact, we will take a deeper look at this in Section 4.5. It can also be indicative that there are some chunks that are unpopular, and thus are difficult to find; this will be explored further in Section 4.4.

4.4 Chunk Popularity

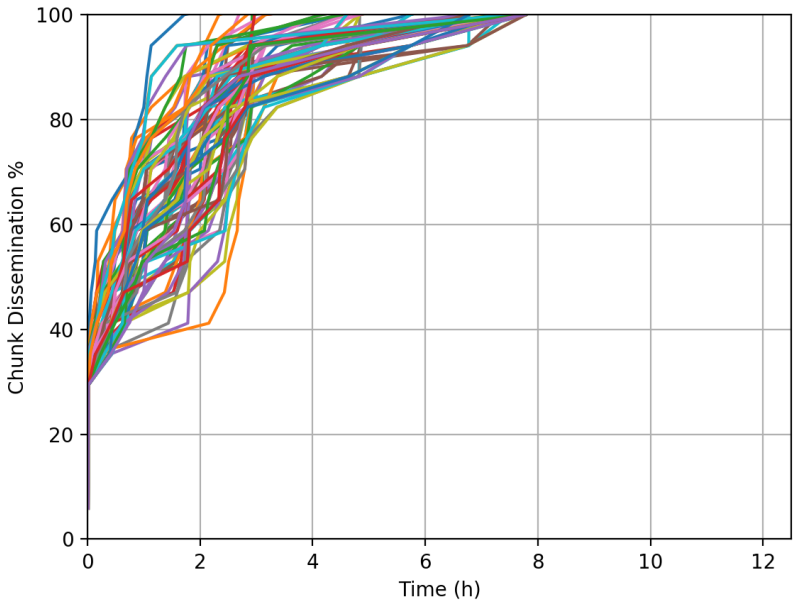
Analysing the distribution of each chunk in the network, Figure 4.3 depicts the percentage of nodes that contain chunk i throughout time. Figure 4.3a and Figure 4.3b clearly show that requesting chunks sequentially causes some chunks to become consistently less popular. This is much more apparent on trial 7 than on trial 6, corroborating the result exposed in Figure 4.2b, and confirming that disparities in chunk popularity can lead to delays in the dissemination of the content. The effects of randomizing the chunk selection are perfectly illustrated by the intense interweaving in Figure 4.3c, that indicates that no chunk was consistently more popular than the rest.



(a) Trial 6 (sequential decision)



(b) Trial 7 (sequential decision)

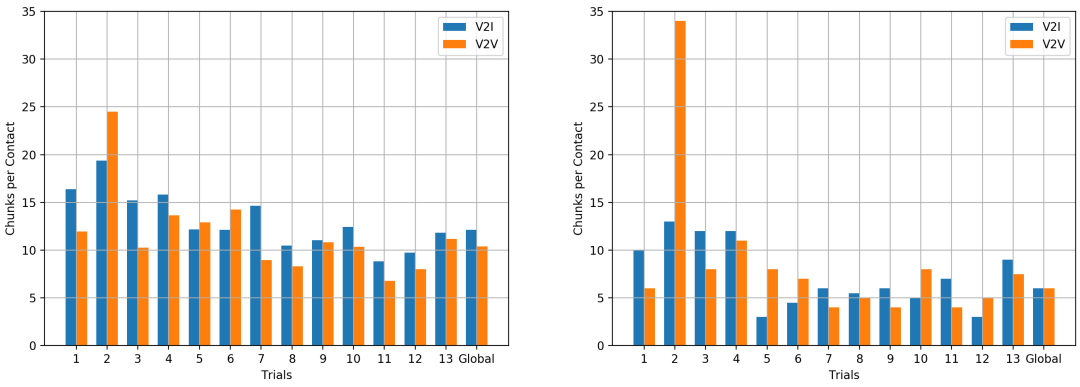


(c) Trial 13 (random decision)

Figure 4.3: Chunk popularity.

4.5 Contact Statistics

In Subsection 3.3.1, the results of some preliminary tests regarding contact durations were presented as a justification for the chosen chunk size. Now, after the deployment of the protocol, it is important to validate these decisions and to look for better insights into the underlying causes for the behavior seen in this specific scenario. With this in mind, Figure 4.4 shows the mean and median of the number of chunks transmitted during single contacts of both V2I and V2V connections. Note that contacts where no chunks were exchanged were removed from this analysis to give a better understanding of the successful communications; failed exchanges will be explored



(a) Mean (b) Median

Figure 4.4: Number of chunks exchanged per contact.

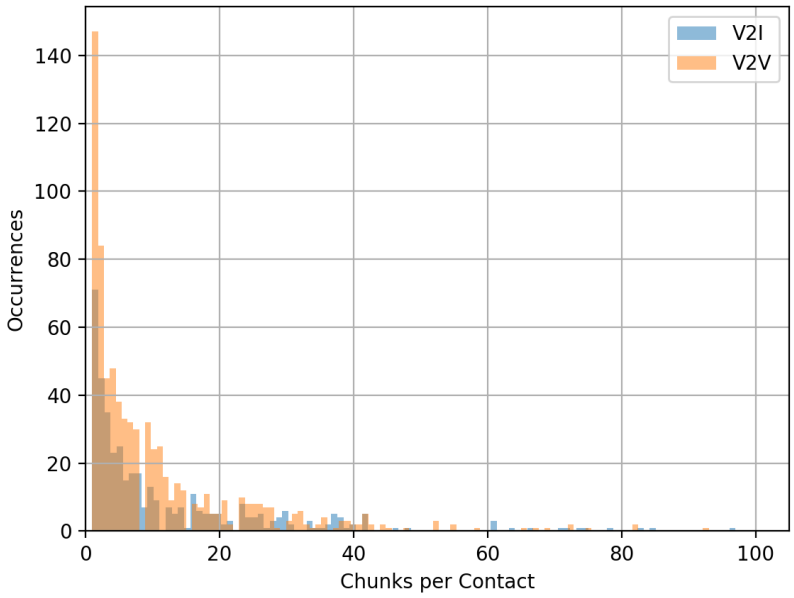


Figure 4.5: Frequency histogram of the number of chunks exchanged per contact

latter in Section 4.7.

Overall, V2I links allowed for more chunks to be transmitted per contact with an average of 12.15 chunks against the 10.41 chunks on V2V connections. However, analysing the median values shows that globally V2I and V2V performed exactly the same with 6 chunks per contact each. These results show that V2I contacts are usually longer than V2V but, otherwise, they are very much comparable.

It is also interesting to note that more than half of the V2V contacts in trial 2 allowed for at least 34 chunks to be exchanged, which corresponds to approximately 81% of the file being transferred in a single contact. This supports the strong correlation between the node completion

and the dissemination evolution seen in Figure 4.2a.

The results presented in Section 4.3 showed that trial 3 had an exceptionally bad performance and that, on that same trial, there were many nodes with incomplete files. Intuitively, one would think that this could be an indication that the contact duration was likely smaller and fewer chunks were transmitted per contact. However, Figure 4.4 shows that this was not the case since contacts that originated complete transfers of chunks were not different from the other trials. This will be revisited in Section 4.7.

Figure 4.5 shows the distribution of the number of chunks transmitted per contact for all trials. It was expected that most contacts allowed fewer than 10 chunks to be exchanged. However, it was not entirely expected that a single chunk would be the mode of the distribution, in both V2V and V2I contacts. This shows that connections are shorter than the expected. Note that this histogram shows the data from 6 trials that had 42 chunks per file and 7 trials that had 105 chunks per file, which is another factor that limits the number of occurrences that could be seen for more than 42 chunks per contact.

4.6 V2I/V2V Importance

Due to the ability of downloading the chunks directly from the cloud, RSUs have the advantage of getting the content before any vehicle and thus the dissemination always starts with V2I communications. Despite the dissemination starting with V2I only, in the long run V2V chunk exchanges play a crucial role in spreading the content across the network, as can be seen by analysing Table 4.2. In all trials, V2V communication accounted for over 40% of chunk transfers. Trial 3 even reached 86.83%, which shows that, on that specific test, the vehicle routes did not cross the RSUs location often.

As stated previously, V2I communications are essential in the early stages of the dissemination as the RSUs are the original seeders of the content. Yet, it is not trivial to understand whether the infrastructure has a special influence or if mobile nodes could be chosen as the original seeders. To better understand this, Figure 4.6 shows the percentage of V2I communications as the dissemination unfolds, i.e. the evolution of the ratio between the number of chunks exchanged through V2I communications and the total number of chunks exchanged. Almost every trial exhibited a behavior similar to trials 5 and 9, where after the initial exchanges, there is a trend for converging to value similar to the ratio of RSUs in the network, which ranges from 25% to 35.7%, as stated

Table 4.2: Percentage of chunks exchanged through V2V communication

Trial	1	2	3	4	5	6	7	8	9
V2V (%)	43.78	54.58	86.83	48.99	65.78	51.20	68.43	55.18	67.52

Trial	10	11	12	13
V2V (%)	50.53	66.44	56.67	56.23

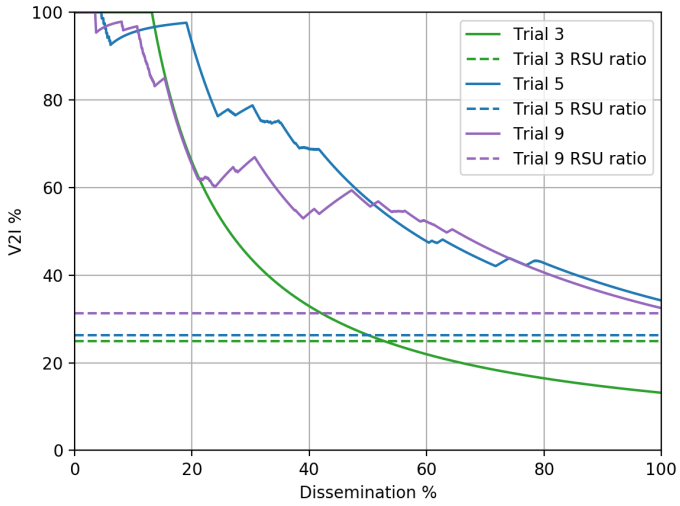


Figure 4.6: Evolution of V2I chunk exchanges

in Section 4.1. This convergence indicates that RSUs are not different from the other nodes and that the same results are expected in networks without infrastructure by selecting some nodes as original seeders. This means that networks can be designed without an infrastructure and content dissemination would perform similarly.

One curious exception to this pattern is trial 3. After the initial 84 chunks have been exchanged through V2I links, all other chunks were downloaded through V2V communication, as can be seen in the hyperbolic curve in Figure 4.6. Again, this might be due to the specific routes taken by the vehicles during the experiment.

4.7 Transmission Failures

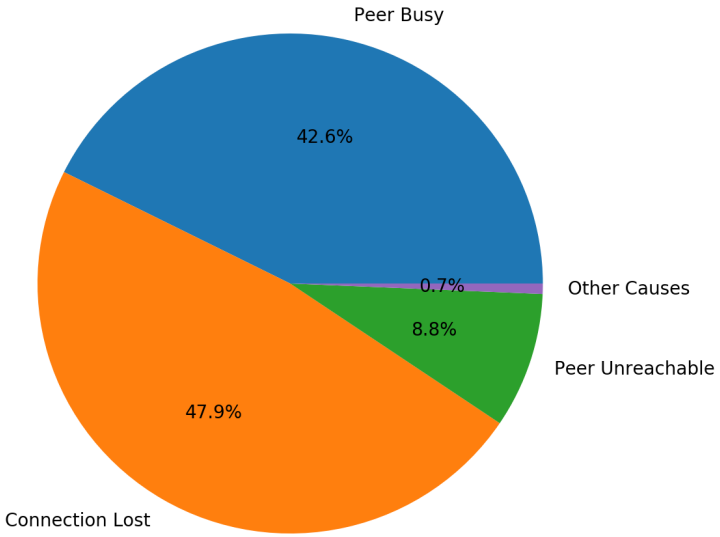
In the 13 trials conducted, a total of 12600 chunks were successfully transmitted and 10006 chunk exchanges failed; this is 44.26% of all download attempts failed, corresponding to a global failure ratio of 79.41%. Table 4.3 shows the failure ratios for each trial; we can highlight that trial 3 had more than twice as many failures as successful transfers, which explains the fact that it performed a lot worse than the others. The high failure ratios degrade the dissemination performance, and as such it is important to dig deeper into their causes.

Download failures can have several causes: (a) the peer is already busy and rejected a new connection, (b) connection is lost during the download, (c) there is no route to the peer, (d) the chunk's hash verification failed, or (e) some other undisclosed error happened. Figure 4.7a shows a global view of these causes while Figure 4.7b provides a per trial analysis. It can be seen that the main causes for a download failure are the peer being busy and connection being lost during transmission. In contrast, there were no failures due to a bad hash, which is expected as this should not happen in normal circumstances.

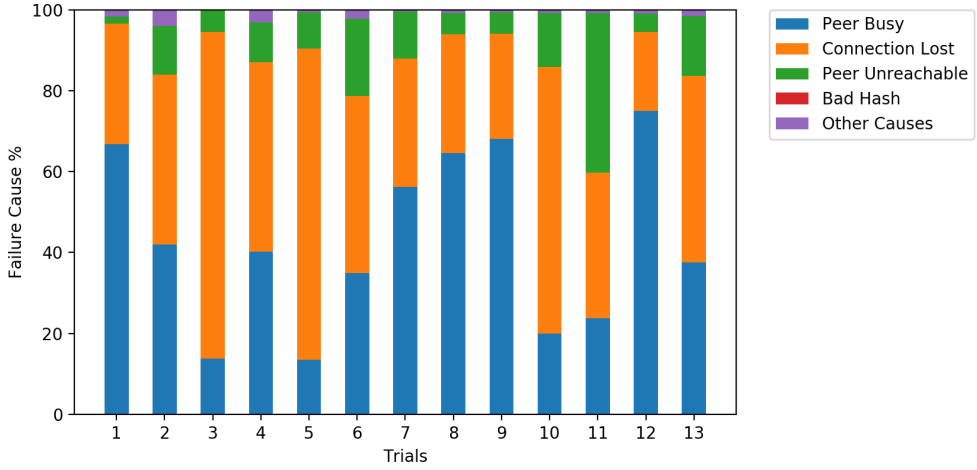
Table 4.3: Failure ratio

Trial	1	2	3	4	5	6	7
Failure Ratio (%)	23.9	6.6	256.3	15.7	65.7	17.5	85.9

Trial	8	9	10	11	12	13
Failure Ratio (%)	63.2	67.2	20.1	37.3	65.9	22.4



(a) global



(b) per trial

Figure 4.7: Failure causes.

Connection loss and the peer being unreachable are problems inherent to the mobility of the nodes in the network. Connection loss happens when a download is interrupted, for example due to weak signal strength. This failure cannot be avoided by the protocol, but its effect can be minimized as will be explained in Section 4.8.2. The unreachable peer failure happens when in the time between a node receiving an availability report and performing the download request a path to that peer is no longer available or valid, e.g. the vehicle moved away. As mentioned in Section 4.1, these tests were conducted using a probing period of 500 ms and a time to live of the availability table records of 2 s. This proved to be a mistake, as the table records should be obsolete after the probing period, because if an answer to the last availability probe is not received, then a chunk download should not be attempted. Therefore, by reducing the time to live of the availability records to a value similar to the probing period, the number of unreachable peer failures should be reduced.

On the other hand, errors related to the peer being busy are not caused by networking issues but by the limited number of seeder threads. This type of failure happens when a node tries to connect to another peer but its connection is rejected, meaning that all seeder threads are already busy sending chunks to another node. Remember from Section 3.4 that the number of seeder and downloader threads can be configured to determine how many upload and download connections are supported simultaneously. Also, all trials were conducted using only one seeder thread and one downloader thread, limiting the number of concurrent links to one upload and one download. By allocating more threads to handle the seeding of the content, multiple simultaneous connections are allowed and, therefore, the amount of rejected requests can be reduced. However, the number of connections allowed needs to be handled carefully as this might increase network traffic, especially when the network is scaling up and large clusters are formed. Also, handling multiple simultaneous connections means that the TCP congestion control mechanisms reduce the available throughput per chunk, and thus the chunk size calculations might need to be adjusted to compensate for that fact. In practice, there might not be a significant difference in overall dissemination time. This type of failures can also be greatly reduced if the seeder node stops answering to availability probes when all its seeder threads are already busy. However, unlike the previous optimization, this does not help the node that is probing because it is simply ignored.

Focusing on the trial 3 again, it was discussed on Section 4.6 that, due to the higher V2V chunk exchange percentage, this specific trial must have had mobility patterns that did not overlap the RSUs position often. The high failure ratio of 256.3 % and the fact that 80.86 % of the failures were due to connection loss, shows that the connections were more volatile in this trial than the rest. Joining the fact, exposed in Section 4.5, that the number of chunks exchanged in successful contacts showed no real difference from the other trials, it can be concluded that most contacts were small enough that no chunks were successfully transferred, while the ones that were successful lasted long enough to account for an average of 10.26 chunks per V2V interaction. Once again, this average does not take into account the contacts where no chunks were exchanged.

This conclusion points to the possibility that some correlation between the ratio of V2V chunk exchanges (Table 4.2) and the transmission failure ratio (Table 4.3) might exist. Even though this

data set is very small, only 13 trials, a Pearson's correlation analysis shows that there is a very strong positive correlation between these two variables, with a coefficient of $r = 0.861$ and a p-value of $p = 1.55 \times 10^{-4}$ that makes it statistically significant for a significance level of 0.01. This indicates that there is a probability of only 1.55×10^{-4} that an uncorrelated system produces data sets that would have this same result. This does not imply that there is a causal link between these two measures, i.e. it is not the existence of V2V chunk exchanges that causes transmission failures or vice versa. However, this indicates that the network conditions that emerge when mobility patterns allow for more V2V chunk exchanges can also induce more failures. Since Pearson's correlation analysis assumes that both data sets are normally distributed and there is no evidence that this is, in fact, true, a Spearman rank-order correlation analysis was also done which produced results that backup the former claims with $\rho = 0.824$ and $p = 5.30 \times 10^{-4}$.

4.8 Overhead

The versatility of the dissemination protocol presented in this work comes at the cost of transmitting excess data that ultimately does not contribute the retrieved content. This wasted data can be twofold: (a) probing overhead which is all the data exchanged with the intention of inferring the content availability of neighbors, and (b) retransmission overhead, which is all the data that is discarded on a transmission failure, i.e. partially-downloaded chunks.

4.8.1 Probing Overhead

The probing overhead metric is important to analyse the impact of using a probing strategy and to compare the two methods tested. As explained in Chapter 3, the probing is done periodically, meaning that the number of sent probes is correlated with each node's download time. Table 4.4 shows the results on a per trial basis and makes it easy to draw this relation between the download time and the number of probes sent. However, the number of probes alone is not indicative of the impact of this strategy as it does not take into account the probe size nor the actual content size; therefore it is important to calculate the overhead it generates. It can be concluded that probing

Table 4.4: Download time and probing statistics

Trial	1	2	3	4	5	6	7
Mean Download Time (s)	2231	3921	13796	4908	13132	7336	21713
Probes Sent	79513	127749	479235	169500	538658	236662	629389
Probing Overhead (%)	0.243	0.368	1.305	0.455	0.702	0.300	0.816

Trial	8	9	10	11	12	13
Mean Download Time (s)	11336	9527	4051	8067	7637	8921
Probes Sent	318141	296711	129594	306550	213812	303302
Probing Overhead (%)	0.380	0.461	0.386	0.769	0.350	0.408

the environment does not significantly impact the amount of data transferred as most trials yielded a probing overhead inferior to 1%. Also, there was not a noteworthy difference between both probing methods, as the range-based method produced a median overhead of 0.455% and the bitmap-based method showed a median value of 0.397%. The probing overhead of the bitmap-based approach can be further reduced by removing the bitmap from the request message, i.e. the data block of the sent probe would just include the file ID and the response message would be the only to include the bitmap.

4.8.2 Retransmission Overhead

As explained in Section 4.7, a chunk exchange can fail due to several reasons. Connection loss failures usually happen after some data has already been exchanged which will then have to be discarded, thus constituting useless overhead. As can be seen in Table 4.5, these failures lead to a substantial overhead with a median value of 2.773% of the total dissemination data and reached a maximum of 4.181% in trial 11. Looking at the overall results, taking into account all trials, there was an overhead of 2.816%. To put these values into context, this overall scenario with a file of 100 MB disseminated across 20 nodes, culminates in 2 GB of data transmitted across the network with 56.32 MB being discarded due to retransmissions. However, not all connection loss failures constitute data overhead as some of theses failures happen in the time frame between the connection establishment and the actual exchange of data. In fact, 65.37% of all connection loss failures happened in this exact circumstance. It is interesting to note that 94.80% of these connection loss failures without any data transferred happened in V2I communication, which is a clear indication of a network congestion problem. Remember that, even though only 17 vehicles were participating in the dissemination trials, this network is comprised of a fleet of over 400 vehicles that share the same DSRC channel and prioritise V2I connections to offload sensor data to the cloud, provide free Wi-Fi for the passengers, among others. Data showed that, in these failures, the connection was successfully established, the downloader successfully sent the request packet to seeder RSU, and that the failure happened in the transmission of the first data packet. The request packet reaches the RSU without a failure because, due to its small size, it has a low collision probability. On the contrary, the first data packet likely has the maximum transmission unit (MTU) size, which takes longer to be transmitted, and thus has a higher collision probability, even with carrier sensing.

Table 4.5: Retransmission overhead

Trial	1	2	3	4	5	6	7
Retransmission Overhead (%)	2.131	0.786	3.234	1.566	2.649	2.261	2.391

Trial	8	9	10	11	12	13
Retransmission Overhead (%)	3.956	3.265	2.773	4.181	2.877	3.168

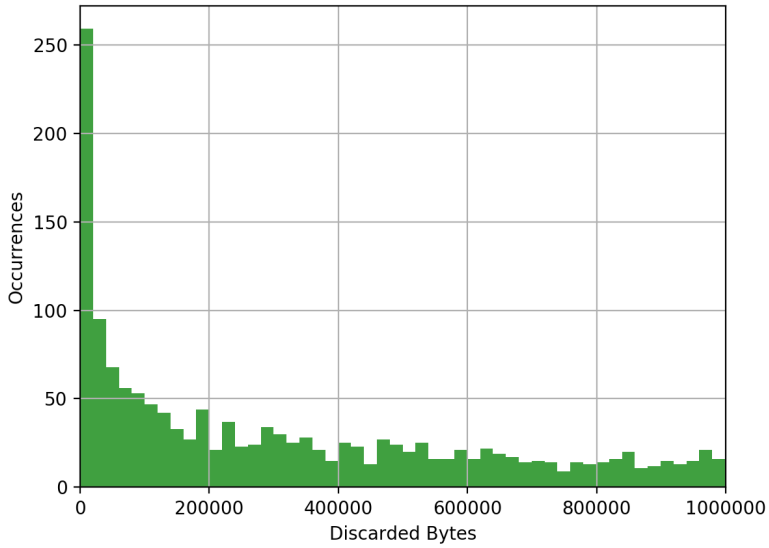


Figure 4.8: Frequency histogram of discarded data due to connection loss

Figure 4.8 shows the frequency histogram of the amount of data discarded per connection loss failure, in all trials; note that the failed transfers with no data loss were ignored for this graph. There is a tendency for failures to happen when there are still small amounts of data exchanged. This is due to the TCP slow start mechanism, where the transmission starts with a small congestion window that limits the amount of data that can be transmitted, and thus the first bytes take longer to transmit. Assuming that the error probability is constant, the fact that the first bytes take longer to be transmitted leads to a higher failure probability. It can be seen, in Figure 4.8, that the probability of a failure happening after 200 kB have been transferred is approximately constant. Nonetheless, these late failures have a big impact on the overhead as they involve more data being discarded.

A rough calculation of the reduction that using a smaller chunk size would have in the overall overhead can be done by assuming that the time between chunk transfers is 0, that there is no data exchange to perform the request, and that the failure would happen exactly at the same point in time. For example, in this approximation, changing the chunk size from 1 MB to 500 kB would imply that a failure happening at a point in time that led to 800 kB of data being discarded would now imply that 500 kB would have been successfully transferred and only 300 kB would have been discarded. Applying this calculation to the data collected shows that reducing the chunk size to 500 kB would imply a reduction of 41.21% in overhead, bringing it to 1.655% of the total dissemination data. Changing the chunk size does not alter the probe size when the range-based probes with sequential chunk decision is in place, like in trials 1-9. However, in the case of bitmap based probes, trials 10-13, the probe size would increase by 21.74% for the 40 MiB files and 41.94% for the 100 MiB file, increasing the probing overhead by the same amount. Given the difference between the results of the probing overhead and the retransmission overhead, reducing the chunk size seems to be a good optimization, although it is still unknown if other factors like the

TCP connection establishment overhead and the request latency would be significantly impacted by the larger number of chunks.

Chapter 5

Conclusions and Future Work

5.1 Synthesis

Over the last years there has been a growing need to move data from the cloud to vehicles in an efficient way. In this work, a data dissemination protocol that focuses on cooperation between nodes has been studied and implemented. The proposed implementation uses V2V communication links to create a cooperative P2P network that successfully disseminates content across the network without resorting to expensive cellular networks. The presented protocol was designed based on the analysis of real world data of Veniam's deployed VANET in Porto but, nonetheless, all implementation decisions were focused on making this work as versatile as possible.

The protocol was successfully implemented in Veniam's production devices and its performance was evaluated in the live service network. Throughout the several trials conducted, it was apparent that the mobility patterns of the vehicles were a decisive factor that heavily influenced the dissemination time. This was not surprising given that it is the dynamic mobility of the nodes that makes vehicular networks a very particular case study. Tests were conducted in a scenario that took advantage of the implemented infrastructure by using RSUs as the original content seeders. However, results showed that V2V data exchanges were more prevalent and it was concluded that, by choosing some nodes to act as the original seeders of the content, similar behaviors can be achieved without the support of an infrastructure. This implies that these initial seeders would need to use cellular data to download the content which goes against the original goal of reducing the cellular data to a minimum but would, nonetheless, greatly reduce the cost of dissemination. It was also seen that some congestion problems were present when using V2I links. Also, two distinct options for the availability probes and chunk selection have been studied. It could be concluded that using a bitmap to represent the missing chunks and basing the chunk decision on a pseudo-random algorithm reduces chunk popularity mismatches; it is speculated that on networks with higher vehicle density real benefits can be achieved by using this strategy.

5.2 Future Work

Due to production constraints, testing was confined to a small network of 17 vehicles and, although some patterns emerged that made it possible to speculate about how this protocol would scale up, it is still unclear how it would behave in a much larger and denser network. Therefore, studies should be conducted to better understand if greater vehicle densities facilitate the dissemination or if this cooperative dissemination model will run into congestion problems. Some congestion problems related to V2I connections have been discussed in Subsection 4.8.2; it would be interesting to see if running similar trials on a DSRC channel separate from the one used in the production network would alleviate the observed effects.

As mentioned in Section 4.7, it was common to see download attempts being rejected due to peers being busy exchanging chunks with other nodes. This shows that, even on this small network, there was a clustering phenomenon were multiple nodes were simultaneously attempting to access the same seeder node. In the same section it was also stated that to improve this situation, the number of allowed simultaneous connections could be increased. However, there is a high probability that these competing nodes are missing the same chunks and thus it could be created a multicast group to transmit the content to several nodes at the same time, increasing the efficiency of the dissemination. This scenario is also adequate for the use of network coding. This can be especially useful in public transportation depots where a large cluster is formed for long periods of time.

In Subsection 3.3.2, it was stated that hash failures might indicate a storage malfunction. This storage problem might be either from the downloader or the seeder node. Therefore, it would be useful to define a set of messages that could be exchanged to notify the other node of this situation. These messages could also be extended to other types of failures and performance issues, in order to help neighbor nodes to optimize the peer and chunk selection process. Also, there are other optimizations that can be made to the selection algorithm. For example, taking into account the mobility of adjacent nodes to choose connections that are expected to have a longer duration or prioritizing chunks that could take advantage of already established connections, thus minimizing the overhead inherent to the creation of TCP streams. The performance evaluation also indicated that some performance gains could be had by optimizing the chunk size parameter. A more comprehensive analysis of how this affects the dissemination would be beneficial to adapt the protocol to different network conditions.

As stated previously, results showed that the dissemination process would behave similarly if, instead of RSUs, some OBUs were selected to download the content directly from the cloud, and thus become the original seeders. Like many studies showed, RSU locations can be a decisive factor in the effectiveness of these networks [3, 4, 5]. Similarly, the decision of which mobile nodes should be selected to be the seeders can also have a tremendous impact in the dissemination. Therefore, choosing a minimum set of OBUs that minimises the dissemination time is a problem left open by this work. Instead of selecting a few OBUs to receive the whole content, it could also be explored the effect of sending just a few chunks to a larger set of OBUs.

References

[1] George Dimitrakopoulos and Panagiotis Demestichas. Intelligent Transportation Systems. *IEEE Vehicular Technology Magazine*, 5(1):77–84, mar 2010. URL: http://ieeexplore.ieee.org/document/5430544/, doi:10.1109/MVT.2009.935537.

[2] Sherali Zeadally, Ray Hunt, Yuh-Shyan Chen, Angela Irwin, and Aamir Hassan. Vehicular ad hoc networks (VANETS): status, results, and challenges. *Telecommunication Systems*, 50(4):217–241, 2012. URL: http://link.springer.com/10.1007/s11235-010-9400-5, doi:10.1007/s11235-010-9400-5.

[3] Evellyn S. Cavalcante, André L.L. Aquino, Gisele L. Pappa, and Antonio A.F. Loureiro. Roadside unit deployment for information dissemination in a VANET. *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion - GECCO Companion '12*, page 27, 2012. URL: http://dl.acm.org/citation.cfm?doid=2330784.2330789, doi:10.1145/2330784.2330789.

[4] Yazhi Liu, Jianwei Niu, Jian Ma, and Wendong Wang. File downloading oriented Roadside Units deployment for vehicular networks. *Journal of Systems Architecture*, 59(10 PART B):938–946, 2013. URL: http://dx.doi.org/10.1016/j.sysarc.2013.04.007, doi:10.1016/j.sysarc.2013.04.007.

[5] Sara Mehar, Sidi Mohammed Senouci, Ali Kies, and Mekkakia Maaza Zoulikha. An Optimized Roadside Units (RSU) placement for delay-sensitive applications in vehicular networks. *2015 12th Annual IEEE Consumer Communications and Networking Conference, CCNC 2015*, pages 121–127, 2015. doi:10.1109/CCNC.2015.7157957.

[6] Car 2 Car Communication Consortium. C2C-CC Manifesto. *System*, page 94, 2007. URL: http://www.car-2-car.org/fileadmin/downloads/C2C-CC{_}manifesto{_}v1.1.pdf.

[7] Elmer Schoch, Frank Kargl, Michael Weber, and Tim Leinmüller. Communication patterns in VANETs. *IEEE Communications Magazine*, 46(11):119–125, 2008. doi:10.1109/MCOM.2008.4689254.

[8] Daniel Jiang and Luca Delgrossi. IEEE 802.11p: Towards an international standard for wireless access in vehicular environments. *IEEE Vehicular Technology Conference*, pages 2036–2040, 2008. doi:10.1109/VETECS.2008.458.

[9] Yunxin Li. An Overview of the DSRC/WAVE Technology. pages 544–558. 2012. URL: http://link.springer.com/10.1007/978-3-642-29222-4{_}38, doi:10.1007/978-3-642-29222-4_38.

[10] Katrin Bilstrup, Elisabeth Uhlemann, and EG Ström. Medium access control in vehicular networks based on the upcoming IEEE 802.11 p standard. *Proc. of the 15th World Congress on . . .*, (3):1–12, 2008. URL: http://www2.hh.se/staff/bettan/Publications/BilUhlStrITS08.pdf.

[11] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Networking Architecture. Technical report, apr 2007. URL: https://www.rfc-editor.org/info/rfc4838, doi:10.17487/rfc4838.

[12] K. Scott and S. Burleigh. Bundle Protocol Specification. Technical report, 2007. URL: https://www.rfc-editor.org/info/rfc5050, arXiv:arXiv:1011.1669v3, doi:10.17487/rfc5050.

[13] Hyunwoo Kang, Syed Hassan Ahmed, Dongkyun Kim, and Yun-Su Chung. Routing Protocols for Vehicular Delay Tolerant Networks: A Survey. 2015, 2015.

[14] Sergio M. Tornell, Carlos T. Calafate, Juan Carlos Cano, and Pietro Manzoni. DTN protocols for vehicular networks: An application oriented overview. *IEEE Communications Surveys and Tutorials*, 17(2):868–887, 2015. doi:10.1109/COMST.2014.2375340.

[15] Christopher Ho, Katia Obraczka, G Tsudik, and K Viswanath. Flooding for reliable multicast in multi-hop ad hoc networks. In *3rd Intl. workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M'99)*, pages 64–71, 1999.

[16] Sathy Main Road. Performance Comparison of Broadcasting methods in Mobile Ad Hoc Network. *International Journal of Future Generation Communication and Networking*, 2:47–58, 2009. URL: https://ssrn.com/abstract=2149417.

[17] Yu Chee Tseng, Sze Yao Ni, Yuh Shyan Chen, and Jang Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless Networks*, 8(2-3):153–167, 2002. doi:10.1023/A:1013763825347.

[18] A. Bujari, C. E. Palazzi, D. Maggiorini, C. Quadri, and G. P. Rossi. A solution for mobile DTN in a real urban scenario. *2012 IEEE Wireless Communications and Networking Conference Workshops, WCNCW 2012*, pages 344–349, 2012. doi:10.1109/WCNCW.2012.6215519.

[19] Amin Vahdat and David Becker. Epidemic routing for partially connected ad hoc networks. *Technical report number CS-200006, Duke University*, (CS-200006), 2000. URL: ftp://ftp.cs.duke.edu/dist/techreport/2000/2000-06.ps, arXiv:1001.3405, doi:10.1.1.34.6151.

[20] Shinpei Kuribayashi, Yusuke Sakumoto, Satoshi Hasegawa, Hiroyuki Ohsaki, and Makoto Imase. Performance evaluation of broadcast communication protocol DSCF (Directional Store-Carry-Forward) for VANETs with two-dimensional road model. *I-SPAN 2009 - The 10th International Symposium on Pervasive Systems, Algorithms, and Networks*, pages 615–619, 2009. doi:10.1109/I-SPAN.2009.65.

[21] Danlei Yu and Young-Bae Ko. FFRDV: Fastest-ferry routing in DTN-enabled vehicular Ad Hoc networks. *International Conference on Advanced Communication Technology, ICACT*, 02:1410–1414, 2009.

[22] Mohamed Oussama Cherif, Sidi Mohammed Secouci, and Bertrand Ducourthial. How to disseminate vehicular data efficiently in both highway and urban environments? *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob'2010*, pages 165–171, 2010. doi:10.1109/WIMOB.2010.5644988.

[23] Wantanee Viriyasitavat, Ozan K. Tonguz, and Fan Bai. UV-CAST: An urban vehicular broadcast protocol. *IEEE Communications Magazine*, 49(11):116–124, 2011. doi:10.1109/MCOM.2011.6069718.

[24] Ozan K. Tonguz, Nawaporn Wisitpongphan, and Fan Bai. DV-CAST: A distributed vehicular broadcast protocol for vehicular ad hoc networks. *IEEE Wireless Communications*, 17(2):47–57, 2010. doi:10.1109/MWC.2010.5450660.

[25] Raul A. Gorcitz, Prométhée Spathis, Marcelo Dias De Amorim, Ryuji Wakikawa, and Serge Fdida. SERVUS: Reliable low-cost and disconnection-aware broadcasting in VANETs. *IWCMC 2011 - 7th International Wireless Communications and Mobile Computing Conference*, pages 1760–1765, 2011. doi:10.1109/IWCMC.2011.5982615.

[26] Min-te Sun, We-chi Feng, and Lai Ten-Hwang. GPS-based message broadcast for adaptive inter-vehicle communications. *Vehicular Technology Conference Fall 2000. IEEE VTS Fall VTC2000. 52nd Vehicular Technology Conference (Cat. No.00CH37152)*, 6:2685–2692, 2000. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=886811, doi:10.1109/VETECF.2000.886811.

[27] Venkata N. Padmanabhan and Kunwadee Sripanidkulchai. The case for cooperative networking. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 178–190, London, UK, UK, 2002. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=646334.758993.

[28] A. Nandan, S. Das, G. Pau, M. Gerla, and M.Y. Sanadidi. Co-operative Downloading in Vehicular Ad-Hoc Wireless Networks. *Second Annual Conference on Wireless On-demand Network Systems and Services*, pages 32–41, 2005. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1383401, doi:10.1109/WONS.2005.7.

[29] Alok Nandan, Shirshanka Das, Biao Zhou, Giovanni Pau, and Mario Gerla. Adtorrent: Digital billboards for vehicular networks. In *IN PROC. OF IEEE/ACM INTERNATIONAL WORKSHOP ON VEHICLETO-VEHICLE COMMUNICATIONS (V2VCOM*, 2005.

[30] Tianyu Wang, Lingyang Song, Zhu Han, Zhaohua Lu, and Liujun Hu. Popular content distribution in vehicular networks using coalition formation games. *IEEE International Conference on Communications*, 31(9):6381–6385, 2013. arXiv:arXiv:1211.2081v1, doi:10.1109/ICC.2013.6655631.

[31] Wei Huang and Liangmin Wang. ECDS: Efficient collaborative downloading scheme for popular content distribution in urban vehicular networks. *Computer Networks*, 101:90–103, 2016. URL: http://dx.doi.org/10.1016/j.comnet.2016.02.006, doi:10.1016/j.comnet.2016.02.006.

[32] J. Yoo, B. S. C. Choi, and M. Gerla. An opportunistic relay protocol for vehicular roadside access with fading channels. In *The 18th IEEE International Conference on Network Protocols*, pages 233–242, Oct 2010. doi:10.1109/ICNP.2010.5762772.

[33] Y. Wang, Y. Liu, J. Zhang, H. Ye, and Z. Tan. Cooperative store-carry-forward scheme for intermittently connected vehicular networks. *IEEE Transactions on Vehicular Technology*, 66(1):777–784, Jan 2017. doi:10.1109/TVT.2016.2536059.

[34] F. Malandrino, C. Casetti, C. F. Chiasserini, and M. Fiore. Content download in vehicular networks in presence of noisy mobility prediction. *IEEE Transactions on Mobile Computing*, 13(5):1007–1021, May 2014. doi:10.1109/TMC.2013.128.

[35] Shabbir Ahmed and SS Kanhere. VANETCODE: network coding to enhance cooperative downloading in vehicular ad-hoc networks. *... of the 2006 International Conference on ...*, pages 527–532, 2006. URL: http://portal.acm.org/citation.cfm?id=1143654{%}5Cnhttp://dl.acm.org/citation.cfm?id=1143654, doi:10.1145/1143549.1143654.

[36] Uichin Lee, Joon-Sang Park, Joseph Yeh, Giovanni Pau, and Mario Gerla. Code torrent: content distribution using network coding in VANET. *ACM MobiShare '06*, page 1, 2006. URL: http://portal.acm.org/citation.cfm?id=1161252.1161254, doi:10.1145/1161252.1161254.

[37] Ming Li, Zhenyu Yang, and Wenjing Lou. CodeOn: Cooperative popular content distribution for vehicular networks using symbol level network coding. *IEEE Journal on Selected Areas in Communications*, 29(1):223–235, 2011. doi:10.1109/JSAC.2011.110121.

[38] Joon Sang Park, Mario Gerla, Desmond S. Lun, Yunjung Yi, and Muriel Médard. CodeCast: A network-coding-based ad hoc multicast protocol. *IEEE Wireless Communications*, 13(5):76–81, 2006. doi:10.1109/WC-M.2006.250362.

[39] JS Park, Ui Lee, SY Oh, M Gerla, and DS Lun. Emergency related video streaming in VANET using network coding. *ACM International Workshop on Vehicular Ad Hoc Networks (VANET)*, 2006:102, 2006. URL: http://portal.acm.org/citation.cfm?doid=1161064.1161087, doi:10.1145/1161064.1161087.

[40] Yong Li, Depeng Jin, Pan Hui, and Sheng Chen. Contact-aware data replication in roadside unit aided vehicular delay tolerant networks. *IEEE Transactions on Mobile Computing*, 15(2):306–321, 2016. doi:10.1109/TMC.2015.2416185.

[41] Seung-hoon Lee, Uichin Lee, Kang-won Lee, and Mario Gerla. Content Distribution in VANETs using Network Coding : The Effect of Disk I / O and Processing O / H. *SECON'08. 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks,*, pages 117–125, 2008. doi:10.1109/SAHCN.2008.24.

[42] C. Partridge, J. Hughes, and J. Stone. Performance of checksums and crcs over real data. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM 1995*, pages 68–76, 1995. Cited By :15.

[43] Roelof Schiphorst, F.W. Hoeksema, and Cornelis H. Slump. *Undetected error probability for data services in a terrestrial DAB single frequency network*, pages 75–84. Number LNCS4549. Werkgemeenschap voor Informatie- en Communicatietheorie (WIC), Netherlands, 5 2007.

[44] Jonathan Stone and Craig Partridge. When the crc and tcp checksum disagree. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '00, pages 309–319, New York, NY, USA, 2000. ACM.

URL: http://doi.acm.org/10.1145/347059.347561, doi:10.1145/347059.347561.

[45] Philip Hawkes, Michael Paddon, and Gregory G. Rose. On corrective patterns for the sha-2 family. Cryptology ePrint Archive, Report 2004/207, 2004. https://eprint.iacr.org/2004/207.

[46] R. Ding, T. Wang, L. Song, Z. Han, and J. Wu. Roadside-unit caching in vehicular ad hoc networks for efficient popular content delivery. In *2015 IEEE Wireless Communications and Networking Conference, WCNC 2015*, pages 1207–1212, 2015. Cited By :12.

[47] T. Hoßfeld, D. Schlosser, K. Tutschku, and P. Tran-Gia. *Cooperation strategies for P2P content distribution in cellular mobile networks: Considering selfishness and heterogeneity*, pages 132–151. Mobile Peer-to-Peer Computing for Next Generation Distributed Environments: Advancing Conceptual and Algorithmic Applications. 2009.