Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Named Data Networking in Vehicular Ad-Hoc Networks: The Support of Push-Based Traffic for Transient, Periodic Data

Christopher Lynch
[redacted]

Supervisor: Professor Vinny Cahill
Assistant Supervisor: Dr. Saqib Rasool Chaudhry

April 30, 2020

A Masters Dissertation submitted in partial fulfilment
of the requirements for the degree of
Master in Computer Science

Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed: [redacted] Date: 30/04/2020

Abstract

The Named Data Networking (NDN) protocol is a promising network/transport layer replacement for TCP/IP when considering that the majority of traffic on the internet today is content; where content is traffic that is not peer-to-peer and maintains its relevance outside of a conversation between two hosts. Therefore, NDN is a content-centric protocol, focused on the desired data instead of a location where the data resides. The architecture of NDN is pull-based, where communication occurs between consumers and producers. A consumer must request the content that it desires and will only accept content that it has requested. Consumers request content by name and any node that contains the content can reply. This behaviour is supported by in-network caching, where data that passes through a node in the network can be cached in the node's content store (CS). NDN also has built-in security, enabling data to live independently of location and producer. NDN supports multicast behaviour by default, nodes will demultiplex data from a producer towards all requesting consumers. Nodes will also multiplex requests for data in the upstream towards producers.

Vehicular Ad-Hoc Networks (VANET) are a subset of Intelligent Transportation Systems (ITS) and refer to Vehicle-to-Everything communication (V2X). VANETs exhibit a set of network conditions for which TCP/IP is ill-suited. The NDN protocol has been identified as a suitable replacement for TCP/IP in VANETs, though previous research has identified issues with NDN in VANETs. Congestion and delay are two issues. The use of a pull-based architecture requires the generation of an interest packet for every piece of data consumed, adding extra overhead to the network. This pull-based architecture also has consequences when the data is transient in nature. Infrequent events, such as safety-critical events, and events that are periodic, invalidating previously generated data, are two examples of transient data. Transient data is time sensitive, which means that waiting for a consumer to request data is undesirable. Push-based architectures might be better suited for disseminating transient data.

This dissertation evaluates the potential benefits of introducing the ability to push transient periodic data in the NDN protocol. A Green Light Optimised Speed Advisory (GLOSA) system is identified as an application where data is generated periodically, which invalidates its previous incarnation. Installed in traffic lights, GLOSA systems inform vehicles of the optimum speed for passing a traffic light during its green phase.

To evaluate the potential benefits of pushing data in the NDN protocol two pushing mechanisms are implemented, unsolicited data and proactive pushing. Unsolicited data refers to nodes eavesdropping and caching packets that they detect. Proactive pushing refers to nodes sending un-requested data into the network that all other nodes within communication range accept into their CS.

SUMO and ndnSim are used to create a large-scale scenario with realistic traffic modelling and network conditions. SUMO is a microscopic and continuous road traffic simulation package and ndnSim is a network simulator used to evaluate experimentation with the NDN protocol. A comprehensive evaluation for varying degrees of vehicle speed, vehicle density, transmission range and data update frequency are undertaken to better understand the performance of pure NDN, unsolicited data and proactive pushing.

The results indicate that pushing transient and periodic data greatly improves network performance when compared to pure NDN. Unsolicited data results, on average, in a 75% decrease in network packets and a 73% decrease in delay. Proactive pushing results, on average, in a 67% decrease in network packets and a 77% decrease in delay.

Acknowledgements

I would like to thank Professor Vinny Cahill for his support and guidance over the course of this project. The weekly meetings, consistent advice and his passion for the subject matter ensured that I always stayed on the right course.

I would also like to thank Dr. Saqib Chaudhry for all the help, tips, examples and guidance provided over the course of the project. There was more than one occasion where a pointer from Saqib proved pivotal in bringing everything together.

I would like to say a big thanks to all my friends and to my girlfriend. Your encouragement and support, the many jokes and questionable adventures, have made my time at Trinity some of the best years of my life

Finally, I would like to say a heartfelt thanks to my family, especially my mom and dad, for backing me every step of the way. Your unwavering support has lead to this point and I am eternally grateful for every opportunity you have afforded me in life.

Contents

List of Figures

List of Tables

Nomenclature

BSS	Basic Service Set
CAM	Cooperative Awareness Message
CDN	Content Delivery Network
CS	Content Store
CR	Content Router
DNS	Domain Name System
DSRC	Dedicated Short Range Communication
FIB	Forwarding Information Base
GLOSA	Green Light Optimum Speed Advisory
ICN	Information Centric Networking
ITS	Intelligent Transport Systems
MANET	Mobile Ad-Hoc Network
NACK	Negative Acknowledgement
NFD	NDN Forwarding Daemon
PCPH	Percentage Cars Per Hour
PIT	Pending Interest Table
P2P	Point to Point
QoS	Quality of Service
RSU	Road-Side Unit
RTT	Round Trip Time
TLS	Traffic Light System
TLV	Type-Length-Value
WAVE	Wireless Access in Vehicular Environments
VANET	Vehicular Ad-Hoc Network
VIP	Virtual Interest Polling
VNDN	Vehicular Named Data Networking
V2V	Vehicle to Vehicle
V2I	Vehicle to Infrastructure
V2X	Vehicle to Everything

1 Introduction

This dissertation investigates the potential benefits gained from pushing data that is transient and periodic in Named Data Networking (NDN) for Vehicular Ad-Hoc Networks (VANET). NDN is a pull-based, content-centric, network-layer protocol that has been identified as a possible replacement for host-centric protocols such as TCP/IP [3]. This is because the majority of traffic in the internet today is content; where content is traffic that is not peer-to-peer and maintains its relevance outside of a conversation between two hosts. Intelligent Transportation Systems (ITS) have been identified as a particular use-case where the content-centric architecture of NDN could improve network performance when compared to TCP/IP [6].

The pull-based architecture of NDN consists of communication between consumers and producers. A consumer must request the data that it desires and will only accept data that it has requested [10]. This pull-based architecture is not well suited to transient data which is time sensitive. The producer of transient data must first wait for a request before forwarding data to a consumer [7].

There are studies that have investigated the benefits of pushing methods for transient data that is infrequent, such as the work published by Muhammad et al. [7], but no research has been identified that investigates the potential benefits of implementing pushing methods into the NDN protocol for data that is transient and periodic in VANETs. This dissertation will investigate the potential benefits gained, or lack thereof, from pushing data that is transient and periodic in Vehicular NDN (VNDN) networks.

The data dissemination methods evaluated in this dissertation are the existing method of unsolicited data that has been requested by a third party and a novel approach called proactive pushing by a producer. Unsolicited data is where a consumer accepts data that it has not explicitly requested. A consumer in the network must still initiate communication with a producer, but the corresponding reply from a producer will be cached by all consumers in the network within range of communication. As a consumer initiates communication, unsolicited data is not a full push-based communication method. Proactive pushing is where a producer will send data into the network as it is created, which consumers will then accept into their cache. As no consumer initiates communication, the proactive pushing method is a fully push-

based communication method. An illustration of the communication models of the unsolicited data and proactive pushing methods can be seen in figure 1.1.

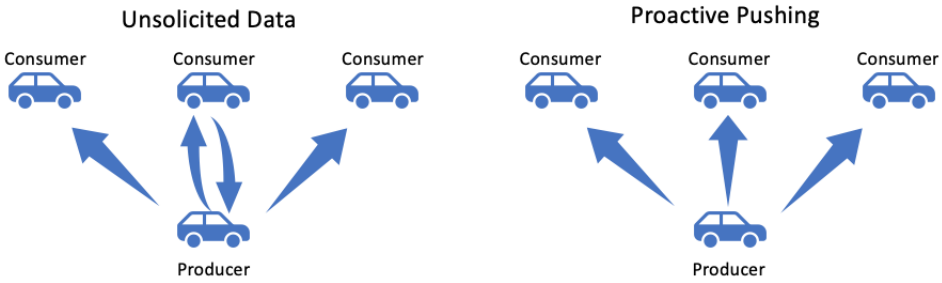

Figure 1.1: Illustration of the communication flow the unsolicited data and proactive pushing methods

1.1 Motivation

Transient data has a limited lifetime before its relevance expires. This limited lifetime means any unnecessary delay in communication is undesirable. This becomes an issue in a pull-based architecture, such as NDN [10], where delay is incurred while waiting for a request to arrive for the transient data before inserting it into the network.

The topology of VANETs is highly dynamic [11], making data exchange between nodes challenging. NDN is a suitable network/transport layer protocol for vehicular environments due to its focus on the *what* of the request instead of the *where*, in-network caching, inherent multicast support and built-in security [12]. But the problem still remains for transient data, a pull-based architecture adds undesirable delay. Also, a pull-based architecture means that for every piece of data consumed there is a corresponding request from a consumer, which can add extra overhead to the network.

There are proposed solutions for mitigating the delay in disseminating transient data in VNDN which introduce push-based methods into the NDN protocol. These solutions for VNDN focus on transient data that is infrequent, such as safety-critical events [7]. There is another type of transient data though. Data which is periodic, meaning, there are frequent updates. One such example of this form of transient data would be a Green Light Optimized Speed Advisory (GLOSA) system, where vehicles can be informed of the optimum speed at which they should proceed in order to pass a traffic light system (TLS) [13]. GLOSA systems produce frequent updates which invalidate previously generated data. The content being produced by a GLOSA system is transient and periodic, as the data is only valid for the duration in which no new update has been produced. In order for a GLOSA system to function using the NDN protocol a polling behaviour would be required. This is where consumers frequently generate requests

for the same information. This polling behaviour for data which is transient and periodic could theoretically produce a lot of congestion, possibly greater than twice the congestion of a push-based behaviour in the same scenario.

The hypothesis is that implementing methods that enable push-based behaviour in the NDN protocol for disseminating content that is transient and periodic in VANETs will reduce the delay in receiving content and reduce the network overhead due to the number of packets being generated. There may also be other improvements such as an increased cache-hit ratio.

1.2 Project Overview

1.2.1 Information Centric Networking

Information Centric Networking (ICN) [14] is a set of guiding principles for a possible future internet architecture based on Named Data Objects. In ICN, content is the fundamental element about which all network functionality is built. The concern focuses on *what* the content is, instead of *where* the content resides. ICN leverages in network caching, multiparty communication through replication, and interaction models that decouple senders from receivers. The goal of ICN is to achieve efficient and reliable distribution of content [14].

1.2.2 Vehicular Ad-Hoc Networks

Vehicular Ad-Hoc Networks [11], derived from Mobile Ad-Hoc Networks (MANET), use vehicles as mobile nodes. VANETs experience a unique set of network conditions such as high speed, short-interconnection times and diverse mobility patterns. Vehicles can quickly move from highly dense, to extremely sparse network scenarios. VANETs also use vehicles as intermediary nodes in the network. For these reasons, specialized protocols have been designed and implemented for VANETs, such as Dedicated Short Range Communication (DSRC) [15] and Wireless Access in Vehicular Environments (WAVE) [5].

1.2.3 Research Aims

The aim of this research is to evaluate the effectiveness of push-based methods in VNDN, for a scenario where the content being requested is transient and periodic. The specific objectives of this research are.

- Implement the unsolicited data and proactive pushing methods in the NDN protocol.
- Design and implement a suitable VANET scenario of an intersection using a TLS exhibiting the communication pattern of a GLOSA system.
- Evaluate the network performance of each method in comparison to pure NDN, as the

density of vehicles, speed of vehicles, transmission range of nodes, and frequency of updates to the transient data change.

1.2.4 Potential Benefits of this Research

The potential benefits of this research include, but are not limited to, a reliable means of disseminating data that is transient and periodic in VNDN. A reduction in delay experienced in the network. A reduction in the number of packets being generated in the network. An overall improvement in network congestion for VNDN from the removal of polling behaviour for data that is transient and periodic. The push-based methods implemented in this project could also apply to any transactions that require polling behaviours, regardless of whether the data is transient.

1.2.5 Project Scope

This project will solely focus on the improvement to the network, measured in terms of congestion, delay, and cache hit ratio, from the implementation of push-based methods in the NDN protocol for the dissemination of content that is transient and periodic in a VANET. The scenario chosen to evaluate potential improvements, or lack thereof, is a four way intersection with a single TLS at the centre of the intersection. This dissertation is not concerned about the potential change in behaviour of vehicles responding to a GLOSA system as a consequence of the various data dissemination methods. Therefore, a GLOSA system will not be implemented but the TLS node will approximate the data dissemination behaviour of a GLOSA system, in so much as can be done under the NDN protocol. This dissertation is also not concerned about any security issues that may arise from the implementation of various pushing mechanisms to the NDN protocol. These concerns would include, enabling eavesdropping from implementing unsolicited data or distributed-denial-of-service attacks from the ability to push data from a producer node. These concerns and more will need to be addressed before the potential implementation of any production ready system.

1.2.6 Road Map

Chapter 2 will provide the background information required to understand the core concepts of this dissertation. This will leads to the state of the art, push-based communication in the NDN protocol. After discussing the state of the art, a brief description of the tools and technology used to evaluate the performance of the pure NDN, unsolicited data and proactive pushing methods will be provided. Chapter 3 will discuss the design and implementation of unsolicited data and proactive pushing. This will include a discussion of the main challenges encountered during each methods development. Chapter 3 will also contain a comprehensive discussion of the design and implementation of the scenario used to evaluate the performance

of each data dissemination method. This will include both the intended implementation and the final design. Chapter 4 provides an evaluation of the results from the simulations running the push-based mechanisms in comparison to pure NDN. This is accompanied by analysis and evaluation of the overall project. Finally, chapter 5 gives a summary of the work done in this project, the results obtained, concluding analysis, and potential future work.

2 Background

This chapter reviews the background information and state of the art research relevant to this dissertation. Firstly, a discussion of the Internet Protocol (IP) and its limitations is provided. Then, an introduction to the NDN protocol and its architecture. This is followed by an introduction to VANETs and then NDN's application in VANETs, which is commonly referred to as VNDN. A brief introduction is then given to the GLOSA system about which testing scenarios will be built. A description of the state of the art research related to the pushing of data in the NDN protocol is then provided. Finally, the tools used to evaluate alterations to the NDN protocol are introduced.

2.1 IP

The internet as we know it, is abstracted into several mutually exclusive layers, sometimes referred to as the OSI stack, allowing the layers and their functionality to develop independently of each other [16]. Interaction between layers occurs via interfaces. The IP protocol, is at the network layer of the OSI stack [17]. Above IP are the transport layer protocols (TCP, UDP, etc.), and the application layer protocols (HTTP, WWW, etc). Below IP is the data link layer where Logical Link Control (LLC) and Medium Access Control (MAC) are performed. Below the data link layer is the physical layer where information is sent through the physical medium. This can be seen in figure 2.1.

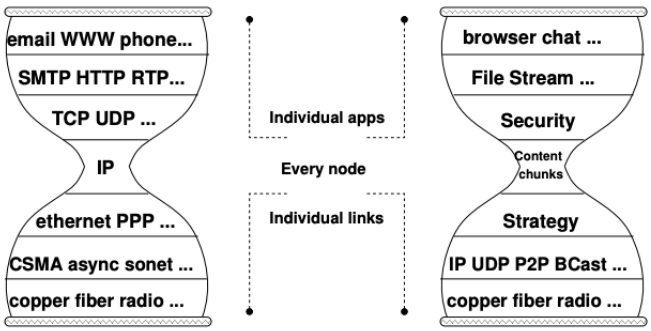

Figure 2.1: OSI model hourglass for IP and NDN from [1]

IP is responsible for providing the necessary information for routing packets across a network. IP uses point-to-point communication between two hosts. It is a conversation oriented architecture for information sharing.

2.1.1 Addressing

IP uses addresses which are represented as numerical labels. There are a finite number of addresses available. For example, IPv4 uses a 32bit address space meaning that there are 2^{32} unique combinations. The finite number of addresses leads to address exhaustion [18], where there are no free addresses left to assign. Address exhaustion can be partially mitigated through the use of address assignment and aggregation strategies such as classless inter-domain routing [19], but these strategies only delay address exhaustion.

2.1.2 Domain Name System

IP requires the Domain Name System (DNS) to function [20]. DNS is a mapping from human readable strings, such as *www.tcd.ie*, to IP addresses. Every time a request is made, a lookup is performed on a DNS server for the IP address matching the request. The purpose of DNS is to save users having to remember IP addresses in order to browse the internet.

2.1.3 IP Packet

An IP packet has a fixed header format, as defined in [17]. The IP header can range from 20 to 60 bytes. The length of the header is specified by the *Internet Header Length* (IHL) field. The minimum viable header size is 20 bytes due to the required fields. The maximum header size is due to the IHL field being 4 bits long, with the length specified in 32 bit words. $32 * 15 = 480$ bits which is 60 bytes. The total length of an IP packet is specified in the *Total Length* field, with the maximum length set by the fact there are 16 bits in the total length field.

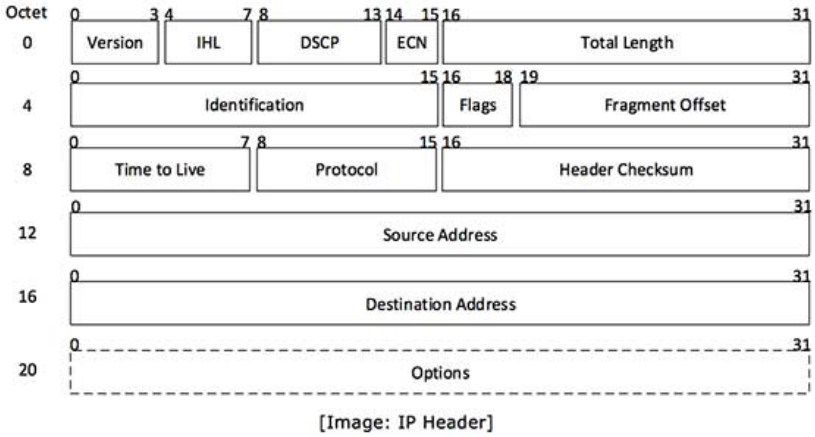

Figure 2.2: IP packet header

2.1.4 Forwarding Information Base

IP uses routing protocols, such as Open Shortest Path First [21], to build its Forwarding Information Base (FIB). The FIB informs switching decisions based on IP address prefix matches. Each match will contain a single outgoing interface. An interface in this case is a network interface. The outgoing interface is the next hop when forwarding a packet towards its intended destination.

2.1.5 The Lack of Caching

An IP router does not perform any in-network caching. When a packet arrives it is placed in a buffer for the duration required to compute the packets next hop. Once this computation is complete, the packet is flushed from the local buffer. The consequence of this is that IP is not built to distribute content across a network. If a piece of content is in high demand, a large volume traffic will flow towards a common point, causing congestion as requests accumulate near a host.

Content Delivery Networks (CDN) are one method that can alleviate the issue of congestion due to popular content. Content Delivery Networks are cache servers setup to distribute data across the globe and move it closer to users [22], as illustrated in figure 2.3. This allows requests for content to be routed towards the most suitable node in a CDN for satisfying the request at that moment in time. According to Cisco VNI [23], as of 2022, 72% of internet traffic will be delivered by Content Delivery Networks. This shows how internet users are increasingly consuming content instead of performing host-to-host communication. Another method of mitigating congestion of popular content is multi-casting. Multi-casting is where a node in the network copies an incoming piece of data and forwards it to all downstream requesting nodes [24].

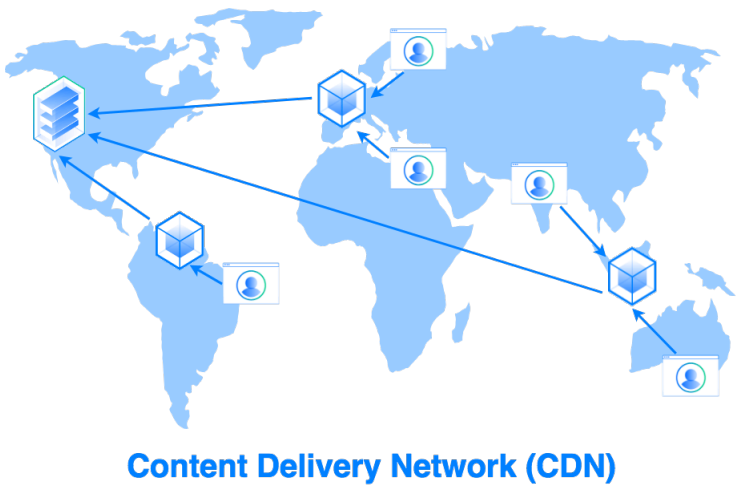

Figure 2.3: Content Delivery Networks as depicted by Ashley John [2]

2.1.6 Security

Security is not originally built-in to IP. The objective was to facilitate communication between heterogeneous networks. As the internet has grown and developed, security has become a prime concern. Security has had to be built on top of IP in response to evolving threats. The suite of security enhancements for IP are referred to as IPSec [25]. Each security enhancement introduces extra overhead and complexity into IP.

2.2 Named Data Networking

To address the deficiencies of IP, the idea of Information Centric Networking (ICN) has been proposed [14]. As the name indicates, in ICN the content is the fundamental element around which network functionality is built. ICN is concerned about *what* the content is, instead of *where* the content resides. ICN is a list of guiding principles, with many realizations such as the Data Oriented Network Architecture (DONA) [26] from Berkley, where DNS is replaced with a data oriented protocol, and the Publish Subscribe Internet Routing Paradigm (PSIRP) [27], which implements the principles of ICN through a publish-subscribe architecture. One of the most popular implementations of the ICN paradigm is Named Data Networking [3] [10], which grew out of Content Centric Networking [1].

2.2.1 Names as Addresses

In Named Data Networking, content is a first-class citizen. Since the focus is on the data itself and not where it is coming from, there is no need for the location-orientated addresses of IP. Instead the name of the data itself can be used as the address. The names are hierarchical but are otherwise arbitrary identifiers. Name semantics are agreed upon by the applications of the producer and consumer nodes and can be any sequence of characters. Names are opaque to the network, with only a delimiter known to a node in order to separate out the hierarchical structure. An example of a name in the NDN protocol can be seen below 2.4.

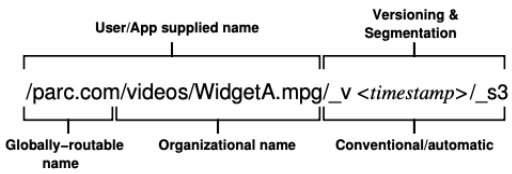

Figure 2.4: Example of name structure as seen in Networking Named Content [1]

2.2.2 Packets

There are three types of packets in NDN, Interest packets, Data packets, and Negative Acknowledgements (NACK). An illustration of the interest and data packet formats can be seen

in figure 2.5.

An Interest packet is created by a consumer and is a request for a piece of data. The packet contains the name of the requested content, *selectors* which are preferences about how the interest and or data packets are forwarded along a route, and a unique identifier called a nonce.

A Data packet is a response to an Interest packet and contains the requested data. It has the content name, which will be used for symmetric routing back to the consumers who requested the data. It also contains meta-information about the data itself, such as freshness period, final block ID and application defined meta-information. It contains the data content, the actual payload that the consumer wants. Finally, the data packet is signed and contains information about the signer.

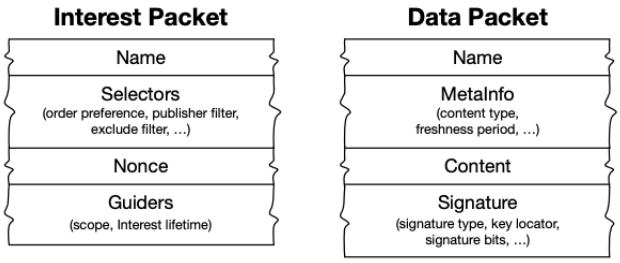

Figure 2.5: Example of Interest and Data packets as seen in Named Data Networking [3] [1]

A NACK packet is a negative acknowledgement. It indicates that a forwarded interest cannot be satisfied. A NACK contains the name of the interest it cannot satisfy as well as an error code indicating the issue. A NACK provides a timely and informative response for an interest that cannot be satisfied.

Each NDN packet is encoded in a Type-Length-Value (TLV) format. The T in TLV indicates the type of block i.e a Name block or Signature block. The L indicates the length of the value block in bytes, and the V is the value of the TLV block. The TLV format allows for a nested structure where the value component is another TLV block. Packets are distinguished by their type number in the first and out most TLV block, TLV_0. A NDN packet is mainly a collection of TLVs inside TLV_0.

The NDN packet format does not have a fixed header or protocol version number. New types can be added and old types removed as required. The absence of a fixed header allows for the efficient support of very small packet sizes, without the header overhead.

As an example, and according to the NDN packet format specification 0.3 [28], a Data Packet would typically be encoded as a Data TLV, TLV_0 (type 6). Inside the TLV_0 block there are another five TLVs, a name TLV (type 7), MetaInfo TLV (type 20), Content TLV (type 21), Signature Information TLV (type 22) and Signature Value (type 23). The MetaInfo TLV has

more nested TLV blocks within. These are Content Type (type 24), FreshnessPeriod (type 25), and FinalBlockID (type 26).

2.2.3 Data Structures

Each NDN node has three data structures, a CS, a Pending Interest Table (PIT) and a Forwarding Information Base (FIB).

The CS is a buffer within a node. As data is routed from a producer to a consumer, the intermediate nodes along the route can cache the data packet being forwarded through them. If a node along that path then receives an interest for the same piece of data, it can satisfy that interest from its CS rather than forwarding the interest upstream to the data producer. Data is distributed through the network in the direction of the consumers requesting it. This form of in-network caching has the effect of reducing congestion at the producer, and improving the locality and availability of data, essentially fulfilling the role of a CDN [29].

The PIT is a record of all interests that have been forwarded by a node but are yet to be satisfied. A PIT entry uses the name of the requested data as its key. This maps to a list of unique identifiers, nonces, for interest packets, and their incoming interfaces, that have requested the data. If an interest arrives at a node that cannot be satisfied by the CS, the node then checks its PIT. If an entry exists in the PIT for that piece of data, then the interface which the interest packet comes from is recorded under the datas entry in the PIT, and the interest itself is discarded. If there is no entry in the PIT for the content name then a PIT entry is created and the interest nonce and incoming interface are recorded.

The FIB is a list of interfaces that can satisfy a request for a piece of data, similar to the IP FIB. When a piece of data is requested that a node cannot satisfy from its CS and there is no PIT entry for the content name, than the FIB is consulted. With each name comes a list of known interfaces that can serve the request. The entries are ranked according to a performance metric determined by the Forwarding Strategy of a node. If no entry exists within the FIB of a node, then the request cannot be satisfied and a NACK packet will be returned. The FIB is populated through the use of a routing protocol. These protocols can be adaptions of the popular protocols used in IP routers today or something entirely new [10][30].

2.2.4 Forwarding and Routing

Forwarding decisions are made by the Forwarding Strategy module running on a NDN node. The Forwarding Strategy is determined by the node owner. Forwarding decisions can be influenced by many factors such as interest packet selectors, the performance of the upstream interfaces, and the wishes of the node owner.

All packets exchanges in the NDN protocol are done through an abstraction called a Face;

short for interface. A consequence of this is that communication between the application layer and the NDN protocol now occurs through an application face instead of through system level calls.

NDN defines Forwarding pipelines which are the steps to be taken when an event occurs. The pipelines for events can be incoming interests, incoming data, outgoing interests, unsolicited data, and incoming NACKs to name a few. The forwarding pipelines define the steps to be taken and the forwarding strategies make decisions as required in a pipeline.

Routing in NDN is symmetric. This means that a data packet follows the same path that was taken by the corresponding interest packet, unlike IP where routing can be asymmetric. This is done via the entries in a PIT. When an Interest packet is to be forwarded, the Forwarding Strategy of the node uses its FIB to determine how to forward the packet. When a Data packet is to be forwarded, it is sent to all interfaces in the PIT that have requested the data. These entries were created by the interest packets as they were routed towards their desired destination. This means NDN has inherent support for multicast operations.

2.2.5 Security

Security is built into NDN. Every Data packet that is produced must be cryptographically signed by the producer [31]. A signature is usually signed by a certificate and this certificate comes from a certificate authority. Any node can be a certificate authority in NDN, supporting trust at all levels [32].

2.2.6 Flow of Data

A brief description of data flow in the NDN protocol will now be given. This flow is also illustrated in figure 2.6. NDN is a pull-based protocol, driven by the consumer. A node will only accept data that it has requested. When a node receives a piece of data, the first operation performed is to check the PIT. If no entry for the received data packet exists in the PIT of a node, than the data will be dropped.

The flow from a consumer to a cache hit for the requested data and back to the consumer is as follows. A consumer creates an interest packet for a piece of data that they wish to obtain. Then, at each node the following operations are performed until the data is successfully retrieved or a NACK is generated.

The CS is checked for the name of the Interest packet. If the data exists in the CS, then it is returned. Otherwise, the PIT is checked to see whether an entry for the requested data already exists. If an entry exists, then the nonce and incoming interface of the Interest are recorded under the requested data name entry in the PIT and the interest is discarded. If there is no entry, then an entry is created in the PIT for the name specified in the interest;

the nonce and incoming interface are recorded. If the interest is not discarded, then the FIB is consulted to determine the optimal upstream interface to forward the interest. If there is no entry in the FIB for the content then a NACK will be returned to all entries for the data in the PIT. Otherwise, the interest will be forwarded towards the ideal interface as determined by the Forwarding Strategy of the node.

When a cache hit for the data is achieved, then a Data packet is created and the following happens. The PIT is checked for entries requesting the data. If one exists, than the data is first stored in the CS of the node and then the data packet is copied and forwarded downstream to all interfaces in the PIT that requested the data. This will continue until the data reaches the consumers requesting it. If no entry exists for the data packet in the PIT of a node, then the node will drop the data packet as it is unsolicited data.

The flow of data in NDN contains well defined and modular components. This gives each node the independence to determine their own behaviour, regardless of the wider network, through altering their implementation and control flow of different components in the NDN protocol. The forwarding strategies allow each node to pursue a wide range of goals. The abstractions of the NDN protocol provide a well-defined baseline to build an organic network from.

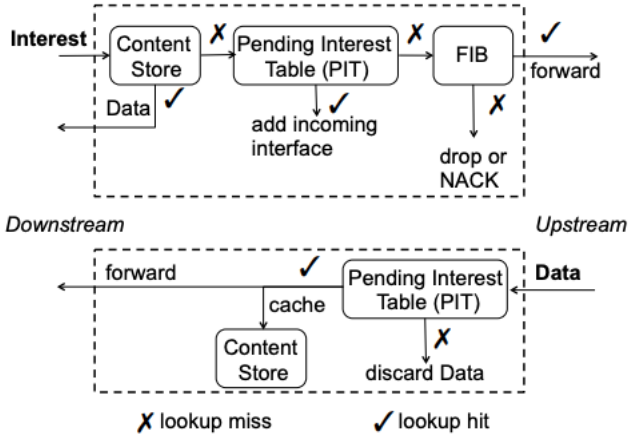

Figure 2.6: Forwarding process at an NDN node [3]

2.2.7 NDN Summary

NDN is a content-centric protocol that follows the principles of ICN. NDN uses names as addresses, which can be hierarchical and are opaque to the network. There are three packets in the NDN protocol, Interest packets, Data packets and NACK packets. The TLV format is used to encode packets. The three core data structures in NDN are the CS, which stores data, PIT, which is a record of interests waiting to be satisfied, and the FIB, which is a list of interfaces that can satisfy requests for data. Forwarding decisions in NDN are made by

Forwarding Strategies and the forwarding steps are contained in Forwarding pipelines. Routing in NDN is symmetric. Finally, Security is built-in to the NDN protocol.

2.3 Vehicular Ad-Hoc Networks

Vehicular Ad-Hoc Networks (VANETs), a sub-domain of ITS [33], refer to Vehicle-to-Everything communication (V2X), from inter-vehicle communication (V2V) to Vehicle-to-Infrastructure communication (V2I) [34]. These forms of communication can be seen illustrated below in figure 2.7. The concept of VANETs has existed for decades; spectrum has been allocated by the US FCC since 1999 for the development of DSRC, intended for use in VANET scenarios [15].

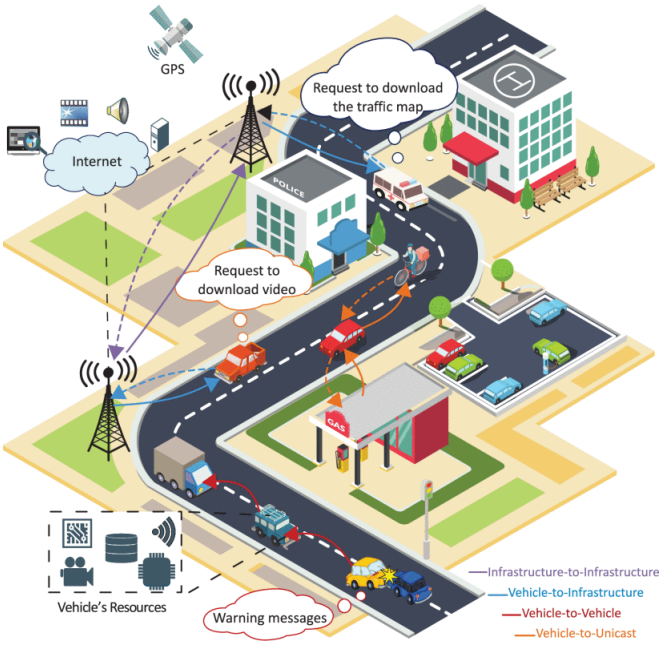

Figure 2.7: VANET as depicted by [4]

VANETs exhibit a unique set of network conditions. Vehicles are the mobile nodes in a VANET which are restricted to the topology of the road. Nodes move at a high speed, in a diverse range of mobility patterns, with short inter-connection times. Nodes can quickly move from highly dense to extremely sparse network scenarios. This is why specialized protocols have been designed and implemented for VANETs, such as DSRC [15] and Wireless Access in Vehicular Environments (WAVE) [5].

2.3.1 Highly-Dynamic Topology

The topology of a VANET is in a constant state of flux; where the VANET itself is restricted by the topology of the road but individual nodes move in a dynamic manner which is influenced by the high speed at which nodes move and the opportunity for unpredictable behaviour that

comes when a human is in the loop [35]. The consequence of this is a short time within which to communicate. In sparse scenarios, there may be no node to communicate with. Conversely, in a dense traffic scenario and urban environments the congestion caused by the volume of packets being requested by nodes has adverse effects on Quality-of-Service (QoS) parameters [36].

2.3.2 Frequent Link Disruption

As a consequence of the high-mobility experienced in VANETs, link disruption is frequent, making routing a difficult challenge. Link disruptions occur when there is opposite-flow traffic, in urban localities, and in non line-of-sight situations. The problem with the first two scenarios is intermittent communication and with the third there are QoS issues due to link deterioration and disruption as a consequence of shadowing. To address these issues and improve connectivity, other nodes in the network can be used as intermediaries. These nodes can be other vehicles or road side units (RSU). A RSU would be a static or mobile access point that can provide connectivity throughout the network, among other functions [37].

2.3.3 Time Constraints

VANETs have time sensitive situations. Safety applications are a time sensitive scenario of particular concern, as human life is at risk [38]. Communication in safety applications much be fast, efficient and reliable in order to give drivers adequate decision-making time. Transient content would be another application where VANETs are delay sensitive. While not safety critical, it is imperative to receive the data before its use has expired.

2.3.4 Wireless Access in Vehicular Environments

WAVE, consisting of the 802.11p [39] and IEEE 1609.4 standards [40], seeks to enable more efficient and effective V2X communication. Changes are made to the Physical and Data Link layers of the OSI stack. The set of WAVE protocols can be seen illustrated in figure 2.8.

802.11p defines a special communication mechanism, enabling operation outside the context of a basic service set (BSS). An infrastructural BSS is a group of 802.11 stations anchored by an AP, think of your home network where a router directs communication. In WAVE the BSS is altered to remove the handshakes and authentication required to join a BSS. This is replaced with the ability to broadcast all the necessary information to join a BSS. Vehicles can then join and leave a BSS as they wish [15].

The wildcard BSSID is also introduced so that all vehicles can instantly communicate with each other if they need to. The BSSID is the unique identifier of a BSS and is the MAC address of the Access Point. For the wildcard, the value is set to all 1s. The wildcard value

can be used for the exchange of critical messages, such as safety messages [15].

IEEE 1609.4 is a MAC layer extension on top of 802.11p. The expressed goal of this protocol is to provide multi-channel operation in the context of a single-radio device for supporting safety and non-safety applications [41]. There are six service channels and one control channel with a channel bandwidth of 10MHz. The 10Mhz frequency is half the bandwidth of 802.11a [42]. As multipath propagation is an important feature in vehicular environments, the added robustness and speed of the 10MHz frequency is an important characteristic.

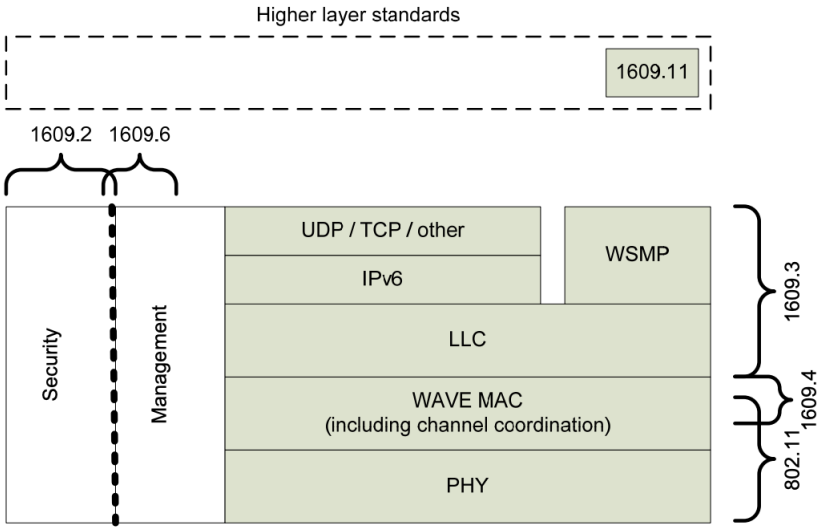

Figure 2.8: IEEE 1609 standards which make up WAVE as depicted in [5]

2.4 Vehicular-NDN

The NDN architecture has been identified as a suitable network/transport layer replacement for IP/TCP in VANETs [11]. It is better suited to a highly dynamic topology and link disruption. NDN can use any network interface available to it. Also, the majority of requests will be for content [23]. An illustration of VNDN can be seen in figure 2.9.

2.4.1 Communication Roles

In the VNDN design by Grassi et al. [6], a vehicle, or node, can assume any one of four roles, a consumer, a producer, a forwarder or a data mule. A consumer is any node requesting content. A producer is a content creating node. A forwarder is any intermediary node through which packets can be routed. Finally, a data mule is a vehicle that is physically carrying content in its CS away from its source.

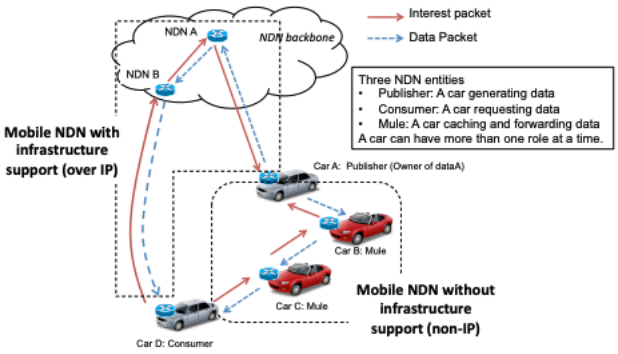

Figure 2.9: Depiction of VNDN as seen in [6]

2.4.2 The Power of Content Naming

Content naming has an important role to play in VNDN aside from content identification. Applications can embed extra information through smart naming semantics. This is displayed in Grassi et als. paper [43], where interests are forwarded by geolocations included in the name of an interest. Focusing on the content, instead of the host-to-host communication found in IP, allows applications to immediately communicate with each other as two nodes come within range.

2.4.3 Caching and Forwarding

In VANETs, it is not always guaranteed that the original content producer will be available. If the original producer is available, there may be issues with link congestion as many nodes direct requests towards the producer. Since a producer and the content they create are decoupled in NDN, content can be distributed across the network. This distribution increases content availability and reduces congestion. In the scenario where there is no connection, data can still be distributed by data muling, making effective use of NDNs in-network caching. NDNs multicast behaviour also serves to improve network performance by aggregating interests and forwarding incoming data to all downstream requests at once. To further this, Yan et al. [12], proposed altering the NDN protocol to include a Data packet aggregation scheme and Interest packet segregation scheme to improve network utilisation efficiency.

Yan et al. [12] also propose to allow unsolicited data caching. This would increase the impact of data muling, increasing data distribution in the network. Unsolicited data is any data that arrives at a node for which there is no corresponding interest. Under the normal tenets of NDN, this unsolicited data would be dropped.

2.5 GLOSA

A GLOSA system informs vehicles of the optimum speed at which they should proceed in order to pass a TLS during its green phase. GLOSA systems have been shown to reduce both $C0_2$ emissions and fuel consumption [44]. Work relating to GLOSA systems dates back as far as 1984 [45] but technological restrictions and low adoption hindered progress. The creation of the 802.11p protocol [39] renewed efforts to implement GLOSA systems [13].

2.5.1 Traffic Light System

In a GLOSA system, traffic lights periodically share their signal phase and timing information with vehicles within communication range. This allows vehicles to compute the optimum speed to pass the TLS during its green phase based on the distance of the vehicle from the TLS and the TLS' signal phase and timing.

TLS' can be static or adaptive. A static TLS follows a fixed sequence of states and transitions indefinitely, remaining in each state for a fixed time. An adaptive TLS can change the sequence of state transitions and the time that it remains in each state, based on information received from external inputs. These external inputs could be information about the number of pedestrians at a crossing or the number of vehicles stopped in a lane. An adaptive TLS controller may perform a signal change with a lead time of 1 second and the state after that change may not be known up until 1 second before it occurs [13].

2.5.2 CAM Packet

The packet type used to disseminate information about the signal phase and timing of a TLS is called a Co-operative Awareness Message (CAM)[46]. CAM packets are used to provide environmental information such as the local road topology or signal phase and timing of a TLS.

2.6 State of the Art in Push-Based Communication in NDN

From the initial papers by Grassi et al. [6] and Yan et al. [12], it is identified that push-based communication in the NDN protocol could be beneficial. Two papers which discuss possible implementations of push-based communication into the NDN protocol are *"Enabling Push-Based Critical Data Forwarding in Vehicular Named Data Networks"* by Muhammaed et al. [7] and *"Internet of Things via Named Data Networking: The support of push traffic"* by Madeo et al. [47].

2.6.1 Enabling Push-Based Critical Data Forwarding in Vehicular Named Data Networks

Muhammad et al. [7] describe the need to implement a push-based method in VNDN for critical-safety information dissemination. It is argued that the pull-based communication model of VNDN proposals to date introduce sub-optimal data forwarding delay, especially when considering critical data that needs to be forwarded promptly.

The paper implements a push-based caching and forwarding mechanism for VNDN that supports pushing content into the network to reduce content forwarding delay. To achieve this, Muhammad et al. implement a beacon packet which is pushed into the network by a producer with critical content. The beacon packet indicates to a consumer that they are about to receive unsolicited data and how many chunks of unsolicited data they are to receive. When a consumer node receives this beacon packet, it creates synthetic interests in its PIT. These synthetic interests are a means of making the data solicited, meaning that a consumer node will now accept the incoming data chunks. The packet flow can be seen below in figure 2.10.

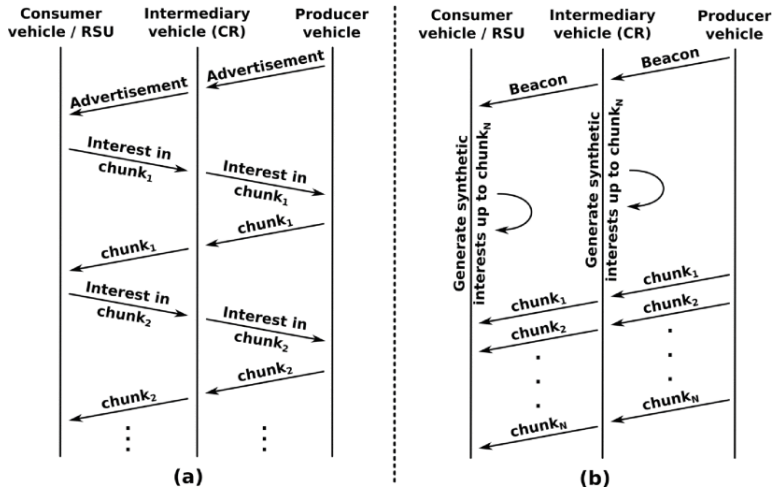

Figure 2.10: Packet flow as seen in [7], where (a) represents traditional ndn and (b) represents Muhammad et als. alterations

The event loop for a producer node is as follows. Any vehicle in the network can be a producer of critical information. A producer will send a beacon towards their neighbouring vehicle or RSU. The beacon contains information about the critical content, such as content object name and size. Beacon packets are altered Interest packets, where an additional field *objectSize* is implemented. The event loop for a producer is shown in figure 2.11. To stop infinite reproduction of the critical data, a destination node TLV is added to the data packet. When the critical data packet arrives at a desired RSU, it will not be forwarded.

Testing was performed using ndnSim (see section 2.7.1), in a scenario where a critical-safety event has occurred. Two types of content forwarding are used, multi-hop and data muling.

PUSH VNDN EVENT LOOP FOR PRODUCER p

Possible events: startup, arrival of interest i from consumer c for chunk r, occurrence of critical event ζ near RSU c.

```
1  case event
2  when startup:
3      advertise each non-critical content object O
4  when i, c, r :
5      send r to c
6  when ζ, c :
7      construct beacon b for ζ
8      send beacon b toward c
9      create and send chunks r_1, ..., r_M toward c
```

Figure 2.11: Pseudocode for RSU or CR as seen in [7]

There is one mobile producer in the network, one RSU destination and several intermediary nodes. Testing was only performed for a single speed of vehicle, so it is not known how the pushing mechanism performs for faster and slower vehicle speeds. Initial performance evaluation showed promising results. The latency for an RSU to receive a critical Data packet was two to three times less than standard NDN. Muhammad et al. conclude that implementing a push mechanism for safety applications in VNDN is promising.

2.6.2 Internet of Things via Named Data Networking: The Support of Push Traffic

In this paper, Madeo et al. [47] consider NDN as a possible replacement for IP/TCP in I.o.T applications. It is identified that certain traits in ICN lend themselves to I.o.T networks, namely easy and scalable data access, energy efficiency, security and mobility support [48]. They also identify undesirable traits between ICN and I.o.T networks, namely I.o.T devices being resource-constrained, the pulling of very small amounts of data and periodic push-based data transmission from monitoring devices. NDN inherently supports the pull-based behaviour of I.o.T networks and so the paper focuses on three possible push mechanisms that can be implemented into NDN for I.o.T networks. The communication flow of each mechanism can be seen illustrated in figure 2.12.

Two I.o.T traffic types that require push behaviour are identified, *Periodic data* and *Event-triggered data*. Madeo et al. define periodic data as as a regular flow of content packets originating from a device towards a control unit. Event-triggered data is defined as an alarm that occurs asynchronously. For example, if the measured blood pressure exceeds a safety threshold, the condition must be timely and reliably reported without solicitation.

Madeo et al. have two main considerations when implementing each push-based communication method, reliability and the scope of the data. Reliability focuses on the countermeasures put in place to cope with the potential loss of packets. Scope of the data is concerned about the suitability of the proposals, given the intended use of the information. Two data scopes are considered, local area traffic, where data is exchanged with nearby devices, and wide area traffic, where data is transmitted over the internet.

The first push mechanism is Interest Notification, designed to support periodic and event-triggered pushing of arbitrarily small chunks of data. This is achieved through including the information in the Interest packet itself as a part of the name component. The idea of interest notification originates from Francois et al. [49]. To ensure reliability, dummy data packets are sent back from the consumer as acknowledgements. This approach does not alter the core tenets of NDN. The use of an Interest packet means the data will not be cached in intermediary nodes.

The second push mechanism is Unsolicited data, proposed for use at a scope of local area environments. Unsolicited data can be used for periodic data or event-triggered contents without any Interest solicitation. When a consumer receives Unsolicited data, they do not immediately drop the packet. First the signature is verified, duplicates are checked for and finally the packet will be admitted into the CS of the node. For reliability purposes, an acknowledgement data packet is then transmitted, the same as the Interest Notification implementation.

The third push-based method is Virtual Interest Polling (VIP), focused only on the periodic pushing of data. VIP uses a concept known as the long-lived interest, conceived in [50]. A long-lived interest is maintained in the PIT of a node for a long period of time. The purpose of this is to allow for the immediate satisfaction of any periodically generated data on a node. The VIP algorithm works as follows. Firstly, a consumer and producer perform a configuration step where producer p and consumer c exchange information about the maximum interval between successive Data generation, the τ parameter. Consumer c then sends a long lived interest and waits for the agreed time interval, vRTO, to receive data. vRTO is defined as τ plus a small safety hysteresis to allow for delay. vRTO is the lifetime of the long-lived interest. When content is received in a timely manner, the long lived interest is simply refreshed. If the vRTO expires before data is received, the consumer sends a regular interest and sets a normal round-trip time (RTT). If a Data packet is not received within the RTT, another interest is sent. Otherwise, on receipt of a data packet, the vRTO timer is newly started with the long lived interest.

The scenario, implemented in Matlab, used to test the three pushing mechanisms is a single consumer producer pair in a local area network with the periodic pushing of data. The two metrics measured are the overhead in the network as a result of the data dissemination method used and the average activity time of a device running each scheme. The overhead to the

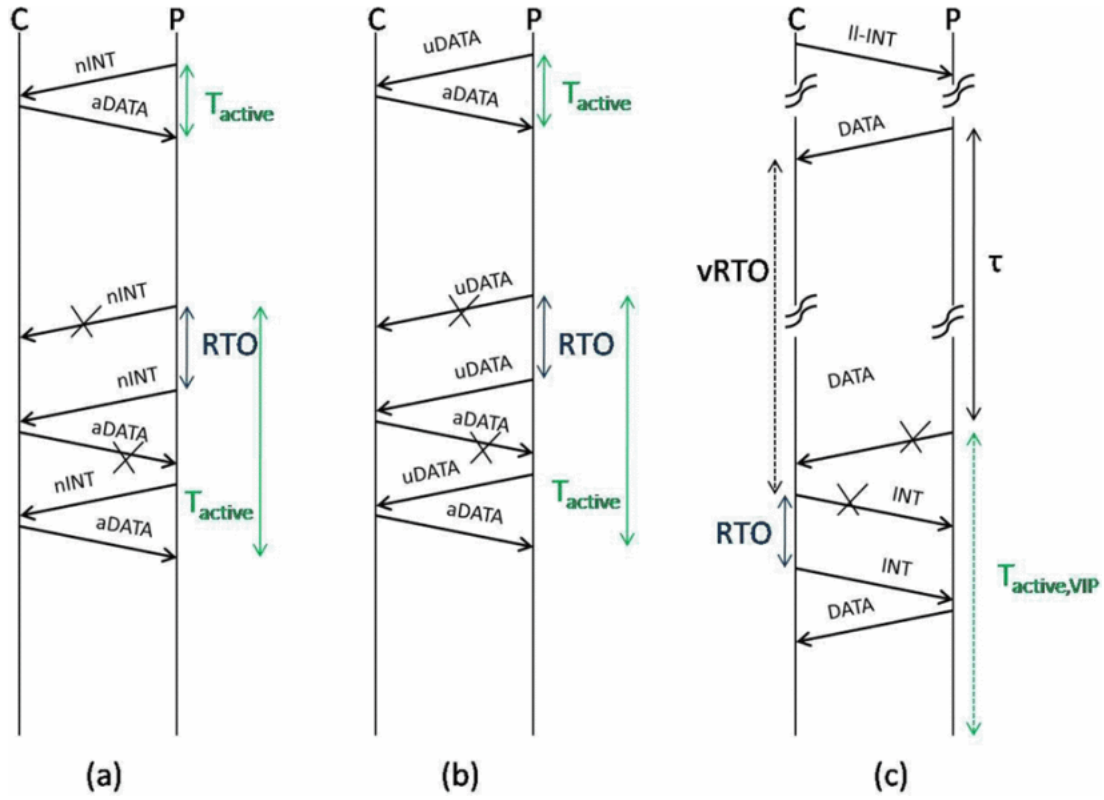

Figure 2.12: Packet flow between a consumer c and producer p for : Interest notification (a), unsolicited data (b), and virtual interest polling (c) as seen in [7]

network is defined as the average number of packets required to successfully exchange a notification. The results for the three schemes showed that Interest Notification and Unsolicited Data had a similar overhead but were almost double the overhead of VIP.

2.7 Tools

This section provides background information about the two tools that were used to develop and evaluate the push-based communication methods in the NDN protocol. The two tools used are ndnSim and Simulation of Urban Mobility (SUMO).

2.7.1 NdnSim

In order to perform experiments with the NDN protocol a means of evaluation is required. This can either be a simulator or a testbed. A testbed is a real-world network which can be used to evaluate alterations to a protocol. The team developing the NDN protocol, in partnership with external institutions, have developed such a testbed [51]. Using a testbed would restrict evaluation to the version of NDN which is installed on the nodes in the testbed network [8]. This reason alone makes it impractical to use a physical testbed as a means of evaluating alterations to the core NDN protocol. The NDN testbed mentioned in [51] is

static and not applicable to evaluating VANETs. It would require considerable resources to establish a VANET testbed for VNDN evaluation, making a testbed infeasible as a means of evaluation for this dissertation.

For quick experimentation with enhancements to the core NDN protocol, it is more desirable to use a simulator. Several simulators have been developed for experimenting with the NDN protocol [52][53], with the preferred simulator for this dissertation being ndnSim [8]. ndnSim is an open source NDN simulator based on the NS-3 simulator [54]. The purpose of ndnSim is to provide the NDN community with a common, user-friendly, and open-source simulation platform.

The functional logic of the NDN protocol is contained in the NDN Forwarding Daemon (NFD) [9] and the NDN primitives that allow for real world experimentation are contained in the NDN-CXX library [55]. NdnSim integrates with both NFD and NDN-CXX which allows ndnSim to provide an integrated simulation environment for researchers and developers to deploy and evaluate their real-world applications and alterations to the NDN protocol at a large-scale. The structure of ndnSim can be seen below in figure 2.13.

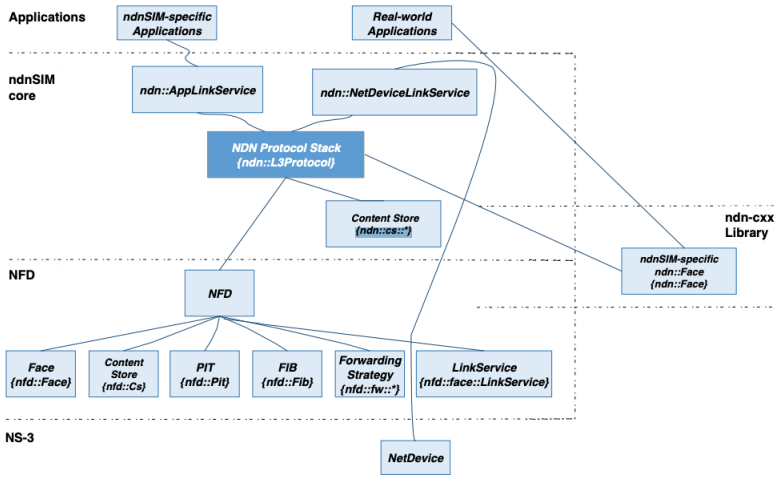

Figure 2.13: Structural diagram of the ndnSIM design components [8]

ndnSim provides researchers and developers with helper classes that allow for the quick and easy configuration of scenarios. NdnSim also provides Trace helpers which simplify the collection and aggregation of various statistical information about the performance of simulations, at all levels, into text files. As ndnSim is integrated into NFD and NDN-CXX, researchers can use the most up to date iteration of the NDN protocol. For these reasons ndnSim is an ideal simulator for experimentation with the NDN protocol and the chosen simulator for scenario creation and evaluation in this project.

2.7.2 SUMO

SUMO is an open source traffic simulation package [56]. SUMO is a traffic simulation software but also a suite of applications which help to prepare and perform the simulation of traffic flows. In order to simulate traffic, two elements are required, road topology and traffic demand.

SUMO is a microscopic traffic simulator, this means that each vehicle and its dynamics are modelled and simulated [56]. This is in comparison to macroscopic simulations where average vehicle dynamics like traffic density and flow are simulated. For SUMO, it is assumed that the behaviour of each vehicle depends both on the vehicles physical ability to move and the drivers controlling behaviour [57]. SUMO's microscopic model is based on the model developed by Stefan Krauß[58].

Road topologies are created using the internal application NETCONVERT or its graphical alternate NETEDIT. NETCONVERT is a command line tool which can import road topologies from different data sources such as OpenStreetMap, OpenDrive, Shapefile or from other simulators such as MATSim and Vissim [56]. NETCONVERT uses heuristic refinement of missing network data to achieve the necessary level of detail for microscopic simulation such as synthesizing TLS' and right-of-way rules. This saves a user having to explicitly define every component of the network.

Traffic demand can be defined as individual trips, flows or routes. The basic information for defining traffic demand is departure time, origin, destination and transport mode such as vehicle or pedestrian. When defining a route, a series of sequential edges to be traversed must be provided. A flow defines the continuous insertion of vehicles into the network, distributed either equally or randomly. These options allow for various levels of fine-grained control over traffic generation in a simulation. Individual trips can be explicitly defined by a user or sumo can insert vehicles into the simulation following a defined flow.

SUMO provides an array of possible output files to allow for quantitative evaluation. These files can be enabled selectively and some possible outputs are vehicle trajectories, traffic data from model detectors and trace files of vehicle paths during a simulation [56].

3 Design and Implementation

This chapter details the design and implementation of unsolicited data and proactive pushing in the NDN protocol for the dissemination of transient and periodic data. The chapter also details the design and implementation of the scenario used to evaluate each data dissemination method. Explanations of the design choices are discussed in detail.

The discussion of the implementation of unsolicited data and proactive pushing is focused on providing a detailed understanding of exactly what is required in order to successfully implement each push-based method. Both from a high level perspective to small architectural details.

The discussion of the implementation of the scenario used to evaluate each method is focused on providing an accurate means of comparison between pure NDN, unsolicited data and proactive pushing. The final scenario implementation allows the hypothesis to be accurately tested; implementing push-based communication in the NDN protocol for disseminating content that is transient and periodic in VANETs improves network performance.

The design and implementation choices lead to the successful implementation of both unsolicited data and proactive pushing into the NDN protocol. A suitable scenario for comparing the performance of each method is also successfully implemented. Enhancements are made to the NFD and NDN-CXX modules to implement unsolicited data and proactive pushing. The implemented scenario is loosely based on a GLOSA system [13] comprising of a single TLS at a four way intersection. The road topology and traffic modelling are created in SUMO. Traffic models for varying vehicle speed and traffic densities are exported as trace files in a format that can be consumed by ndnSim. The trace files describe the path taken by each vehicle over the duration of a simulation. NdnSim defines the network scenario used to generate results for the comparison between the different data dissemination methods.

3.1 Requirements

- Implementation of unsolicited data and proactive pushing to the NDN protocol.
- Implementation of a realistic scenario where the content in demand is transient and

periodic.

- The scenario network conditions should have four configurable *parameters*. Allowing for different configurations of vehicle density, vehicle speed, node transmission range and data update frequency to be evaluated.
- The proactive pushing method should forward data beyond its single-hop neighbour.

3.1.1 Parameters

As stated in section 3.1, the ability to vary the vehicle density, vehicle speed, node transmission range, and data update frequency are required. These parameters are chosen to alter the network conditions of a simulation, allowing for a more comprehensive evaluation of each data dissemination method.

Percentage Cars Per Hour (PCPH) defines the number of cars that use a lane per hour. PCPH can be used as a measure of capacity for a given section of road. The Highway Capacity Report [59] is used for vehicle density measures in this dissertation. The report defines lane capacity as *"the maximum number of vehicles that can pass a given point during a specific period under prevailing roadway, traffic, and control conditions. This assumes that there is no influence from downstream traffic operation, such as the backing up of traffic into the analysis point"*. The PCPH of a lane should affect the volume of requests experienced in a geographic location.

Vehicle speed affects the length of time in which communication can occur between two nodes in a network. It has been shown that in high mobility situations, network performance is impacted by vehicle speed [60]. Therefore, it is important to test the performance of each pushing mechanism over a range of vehicle speeds.

Transmission range impacts the effectiveness of data dissemination. Protocols have been developed to make the best use of available transmission range in VANET environments [61]; showing that varying the physical distance at which a signal can be registered by a nodes receiver impacts network performance.

There should be a positive correlation between the frequency of updates for transient data and the number of packets in a network. This assumes that the frequency of updates matches the request frequency of nodes in the network. As the frequency of updates increases, the number of packets should increase. This could add extra overhead to the network, increasing congestion, and so the effectiveness of each data dissemination method should be evaluated as the update frequency of nodes increases.

3.1.2 Scope of the Data

The scope of the data refers to the geographic area in which the data has meaning, as discussed in section 2.6.2. The scope of the data will impact the requirements for disseminating information in a VANET, such as whether it is possible to route the data. If data is of a global scope then it is important that it be routable. In the case of a GLOSA system the data only holds significance within its immediate geographic area; the paper by Tielert et al. [44] found communication ranges greater than 600m to not be useful in GLOSA systems. Therefore, it can be argued that the data has a local scope, removing the necessity for it to be routable.

While the data is not required to be routable, the original design for proactive pushing had a desire for it to be forwardable. Forwardable refers for the ability of a piece of data to continue propagating throughout the network beyond its immediate single-hop neighbour. This means the data is not following a symmetric route to a consumer but is still being forwarded, on a hop by hop basis, throughout the network. There are two requirements for this to be possible. The first requirement is that the data must not flood the network, ideally only reaching each node once. The second requirement is that the data should not go beyond its area of significance.

The first requirement is satisfied by the CS of a node checking whether it has already received a piece of data. If the node has the data in question in its CS then it ignores any new incoming data. If not, then the data is stored in the CS of the node and forwarded following the original algorithm described in section 3.3.

Two possible approaches were identified for the second requirement. In the first approach, data could have geographic information stored in its name, similar to the method described by Grassi et al. [43]. This geographic information could then be used to determine the distance of the data from the producer on a hop by hop basis. Each node has a probability of forwarding a pushed piece of data that is inversely proportional to the distance from its point of origin, $p \sim \frac{1}{d}$, similar to the method described by Xia et al. [62].

In the second considered approach, the data is given a lifetime, called a freshness period. The concept of a freshness period already exists in the NDN protocol and is implemented in NFD by default. When a piece of data is created by a producer, a freshness period TLV is encoded into the Data packet as a sub-block of the meta-information TLV. When a piece of data arrives at a node which satisfies the criteria to be cached, its freshness period is converted into a timestamp that determines the point in time at which the data becomes stale. Stale data is still valid data and it is up to the caching policy of a node to determine if the data will be removed when it becomes stale. This means that the application has to specify that the interests it creates should be satisfied by fresh data, as there is no guarantee that a nodes cache replacement policy has removed stale data. This is achieved by setting the *MustBeFresh* field of an Interest packet. When this boolean field is set and encoded in a Interest, it is only allowed to be satisfied by data whose freshness period has not expired.

The final design implements the second considered approach for limiting the range of a data packet. In the case of periodic data, it is known exactly when a piece of data will become stale; when new data is generated. If a producer is creating a new piece of content every second, then the freshness period of that piece of data is also one second. The GLOSA application for consumers in the network sets the *MustBeFresh* field of the interest packets it creates and the producer sets the freshness period to match the frequency that it generates new updates. Another factor to be considered is the amount of divergence from the NDN protocol, this should be kept to a minimum and is done so in this case by using the pre-existing freshness semantics.

3.2 High-Level Overview

The core components of the solution are shown below in figure 3.1. The road topology and traffic demand characteristics are defined using XML statements that are interpreted by NETCONVERT to produce a network that can be read by the SUMO application. SUMO then generates a traffic simulation. From the traffic simulation a trace file is output which can be read by ndnSim. The trace file is used to define the path that nodes will follow in the ndnSim simulation. The NFD and NDN-CXX modules are enhanced with two additional methods of information dissemination, unsolicited data and proactive pushing. The scenario is defined using ndnSim through the use of various helper classes. With the trace file, ndnSim scenario and adjustments to the NDN protocol, testing can occur to evaluate the effectiveness of pure NDN, unsolicited data and proactive pushing in disseminating data that is transient and periodic.

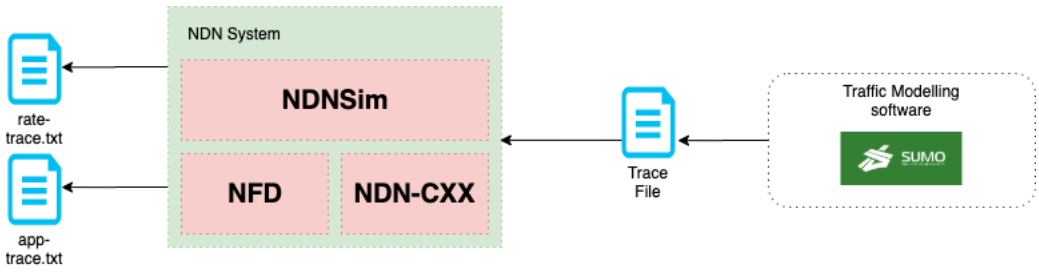

Figure 3.1: High Level implementation overview

The core components can be broken down into three areas for discussion. Firstly, the alterations required to the NFD module to implement both unsolicited data and proactive pushing. Secondly, the alterations required to the NDN-CXX module to implement proactive pushing. Finally, the steps required to correctly configure a suitable simulation for evaluating pure NDN, unsolicited data and proactive pushing.

3.3 NDN Forwarding Daemon

NFD is the network forwarder that implements the core functionality of the NDN protocol. The design of NFD emphasizes modularity and extensibility to allow for easy experimentation with new protocol features, algorithms and applications [9]. The main functionality of NFD is to forward Interest and Data packets. An overview of the modules in NFD are shown in figure 3.2.

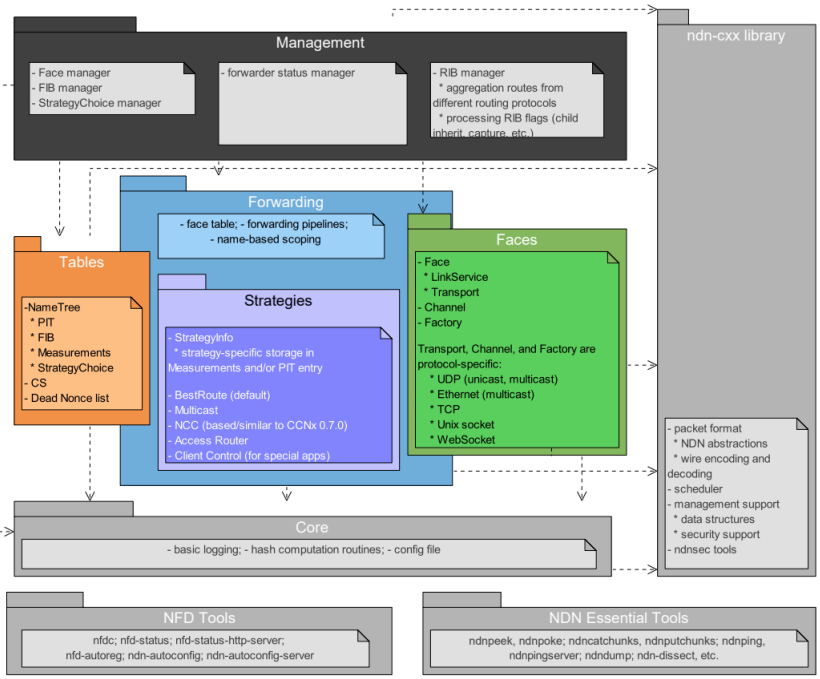

Figure 3.2: Overview of NFD modules and dependencies [9]

To understand the changes that are required in order to implement unsolicited data and proactive pushing, it is best to have an understanding of the modules and their significance in NFD. In particular the Forwarding module where the pipelines, that were discussed in section 2.2.4, are expanded to include pushing and acceptance of unsolicited data.

The core packet forwarding logic is implemented in the Forwarding module. Here the logic of forwarding is broken down into two sub-modules, forwarding pipelines and forwarding strategies.

Forwarding pipelines define a series of steps that are to be taken for each packet. For example, if a node receives a Data packet, it is first inserted into the *OnIncomingData* pipeline. The first operation performed is to check if the node contains entries in its PIT for the arriving data. All matched entries are then selected for further processing. If no entry exists in the PIT, then the data is sent to the *OnUnsolicitedData* pipeline and dropped. In the case that matches are found, the data is added to the CS of the node and the forwarding strategy of

each selected entry is notified of the arrival of the data. This is because NFD is stateful, it will track the performance of each route that interests can be forwarded down, storing performance metrics in a measurements table. This information is used to influence future forwarding decisions of the Forwarding Strategies for each namespace. Where a namespace is the section of a data name that is an identifier for a grouping of related data. The final step is to send data to all downstrean interfaces, recorded in the PIT, through which interests for the data arrived. It is common for packets to be handed from one pipeline to another as conditional checks on a packet are performed. This is seen in figure 3.3.

Forwarding strategies provide the intelligence to make decisions on whether, when and where to forward interests. Forwarding strategies are invoked from forwarding pipelines as a packet is processed. Forwarding strategies also receive metrics about the outcomes of forwarding decisions. These outcomes will be used to inform future forwarding decisions. Forwarding strategies can be invoked on a per namespace basis allowing applications to use the strategy best suited for their intended function. For example a file sharing application may want to use paths with the highest bandwidth whereas a web call application may want the path with the least delay [9].

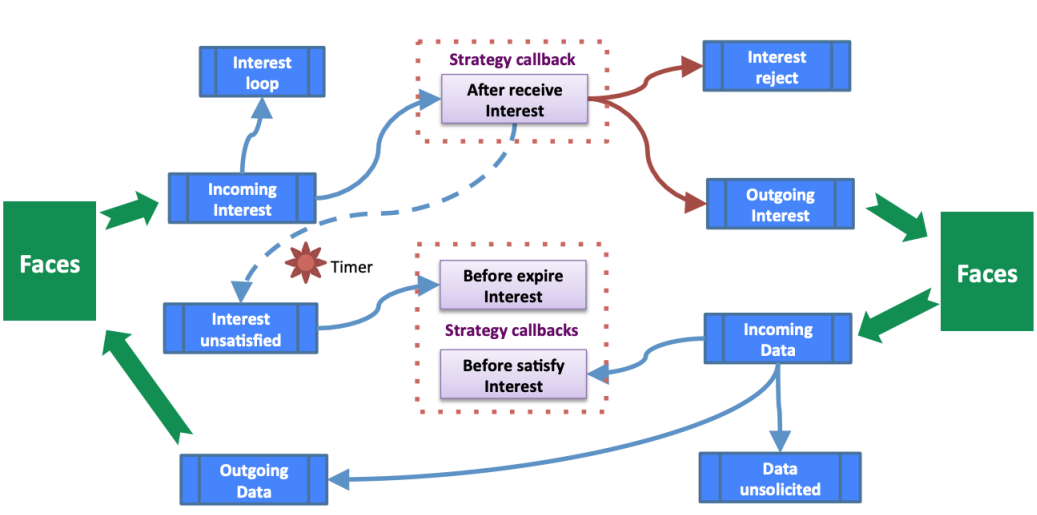

Figure 3.3: Overview of NFD pipelines [8]

The business logic of the proactive pushing and unsolicited data algorithms are implemented as their own pipelines within the forwarding module of NFD. In the base NDN forwarding module the *OnUnsolcitedData* pipeline exists and is entered from the *OnIncomingData* pipeline. The same control flow is used to enter the proactive pushing pipeline that is created for this dissertation; discussed in more detail in section 3.3.1. No alterations are required to implement the desired *OnUnsolicitedData* behaviour, instead the *ndn::UnsolicitedDataPolicy* class will be altered, discussed in section 3.3.2.

Both methods could be implemented in the OnIncomingData pipeline, but this would harm

the future modularity and extensibility of the forwarding pipelines. Instead the unsolicited data and proactive pushing methods should have their own pipelines. The pipelines are invoked from the OnIncomingData pipeline, as a results of conditional logic invoked by the state of an arriving data packet, which is discussed in section 3.4. The updated branching options for the OnIncomingData pipeline can be seen in figure 3.4.

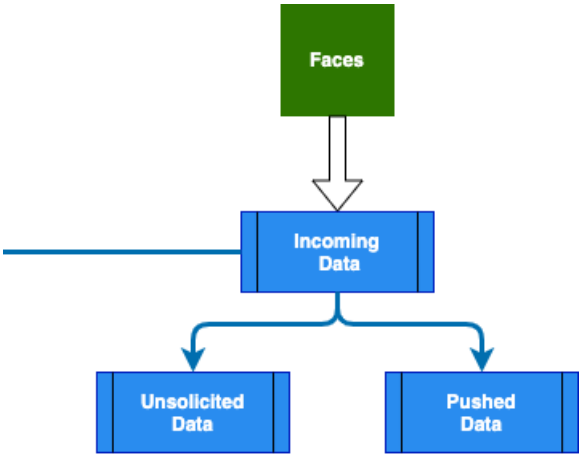

Figure 3.4: Branching options from *OnIncomingData* pipeline after adding the proactive pushing pipeline, Pushed Data

3.3.1 Proactive Pushing Pipeline

The proactive pushing pipeline, referred to as the Pushed Data pipeline, is invoked from the *OnIncomingData* pipeline, as mentioned in 3.3. When a piece of data enters the Pushed Data pipeline, the CS of a node will be checked to see if it already contains the data. If it does, then the data is ignored, otherwise the data is inserted into the nodes CS. This check occurs to stop data flooding the network, satisfying the first requirement for forwarding, as discussed in section 3.1.2.

If data was inserted into the CS of a node, it should be forwarded. Since NDN uses symmetric routing, as mentioned in section 2.2.4, data follows the exact path taken by a corresponding interest packet; this is not possible with pushed data. To enable forwarding of pushed data, a node must know where it wishes to send data. Pure NDN will perform a PIT lookup to discover the faces data should be forwarded to in order to satisfy requests, as discussed in section 2.2.4. In the case of pushed data, there will be no PIT entries. Instead the FIB is used to direct forwarding, as would be done for an interest packet. A FIB lookup performs a longest prefix match on the name of the data packet and returns a list of possible next hops. To ensure a next hop is returned, the producer application configures the FIB with a network interface that can broadcast data to be pushed, discussed in section 3.7.3. The pipeline then determines the lowest cost next hop. The cost of a hop is determined by the Forwarding Strategy for a namespace or manually configured on startup. When the lowest cost hop is found the data is pushed to that Face.

Algorithm 3.1: Pushed Data pipeline

Input: Data

```
1  if !(Data in CS) then
2      Insert Data into CS ;
3      Get possible next hops for Data from FIB
4      for Possible next hops do
5          if hop.cost < minCost then
6              minCost ← hop.Cost;
7          end
8      end
9      for Possible next hops do
10         if hop.cost == minCost then
11             Send data to hop;
12         end
13     end
14 end
```

It should be noted that when a packet arrives at a node, the NDN protocol sets the cost of the face to 2^{31}. This is a check in the NDN protocol to stop any packet being forwarded back through the face that it came through. Nodes are only aware of one network face to push data through. The consequence of this is that pushed data will always be forwarded up to the application layer on arrival at a node because the application face will have the lowest cost. If pushed data is to be forwarded then it will have to be done so at the application layer in the same way a producer would push data. This created another issue, the data that is sent back down from the application layer to be further forwarded will be ignored. This is because it is still marked as a pushed data packet and will therefore enter the pushed data pipeline. Here is will fail the first conditional check as the data will already be contained in the CS of the node.

These issues were discovered late in the development cycle of this project and there was insufficient time to implement a fix that would allow the pushed data to be forwarded through the network. One possible solution is to make nodes aware of many network interfaces to forward the pushed data to as well as the application face. Another potential solution is to have separate control flows for incoming and outgoing data. This can be achieved by checking the type of face that the data has arrived from. If it is an application face, then the data is outgoing and should ignore the *Data in CS* check. If the data has arrived from a network face, than the data is incoming and should be subjected to the CS check.

3.3.2 Unsolicited Data Pipeline

The pipeline for unsolicited data is invoked from the *OnIncomingData* pipeline. Each node has a *UnsolicitedDataPolicy* which is consulted when deciding whether an incoming piece

of unsolicited data will be accepted in the CS of a node. The policy is by default set to *DropAllUnsolicitedDataPolicy*, but can also be one of *AdmitLocalUnsolicitedDataPolicy*, *AdmitNetworkUnsolictedDataPolicy* or *AdmitAllUnsolicitedDataPolicy*.

AdmitLocalUnsolicitedDataPolicy will accept unsolicited local data packets. A local data packet is any packet generated under the "localhost" prefix. This is used for communication between the NFD instance of a node and the applications running on a node.

AdmitNetworkUnsolictedDataPolicy will accept unsolicited network data packets. A network data packet is any packet coming from a network interface on a node.

AdmitAllUnsolicitedDataPolicy will accept all unsolicited data packets. This is a catch all that encompasses the *AdmitLocalUnsolicitedDataPolicy* and *AdmitNetworkUnsolictedDataPolicy* unsolicited data policies.

The desired type of unsolicited data is network packets. Therefore, to enable the correct unsolicited data, the *AdmitNetworkUnsolictedDataPolicy* needs to be selected. This is achieved through altering the NFD source code with a default of *AdmitNetworkUnsolictedDataPolicy*. This would not be sufficient for a production system where different applications will require different caching behaviour but is suitable for the purpose of evaluating the performance of unsolicited data against pure NDN and proactive pushing.

The pipeline itself is a check of the *UnsolicitedDataPolicy* of the NFD instance running on a node.

Algorithm 3.2: Unsolicited Data pipeline

Input: Data

```
1 decision ← UnsolicitedDataPolicy.decide(Data);
2 if decision = cachethedata then
3 |   insert data into CS;
4 end
```

3.4 NDN-CXX

NDN-CXX is a C++ library implementing primitives for NDN. It is used by NFD for an implementation of the TLV format, encoding, security, data packets, interest packets, face abstraction and more. To stay within scope, only primitives requiring alteration in the implementation of pushing mechanisms will be discussed, namely the Data packet, meta-information and TLV encoding primitives.

As mentioned in section 3.3, how incoming data packets are processed depends on conditional logic evaluated based on the state of the packet. In order to implement a proactive pushing mechanism into the NDN protocol, it must be possible to infer whether a data packet has been

pushed. This is achieved by adding a flag to the Data packet primitive, indicating whether a packet has been pushed.

A Data packet is simply a collection of TLVs under TLV_0, which indicates the type of packet, as discussed in section 2.2.2. A data packet is comprised of its name, meta-information, content and signature. A data packets meta-information block is a sub-TLV consisting of Content Type, Freshness Period, Final Block ID and additional application meta-information. This is shown in figure 3.5

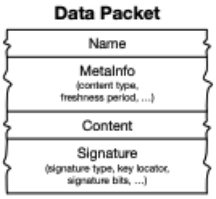

Figure 3.5: Data packet as seen in [10]

3.4.1 Meta-Information

A data packet being pushed can be represented by a boolean flag that adds information about the state of the packet. As being pushed is additional information about the state of a packet, the pushed member variable is implemented in the meta-information primitive and not the Data packet itself. Adding a pushed flag to the meta-information block is not a simple process and any misstep will cause the packet to be incorrectly encoded and decoded, losing information.

The final design includes the addition of a flag indicating that a data packet is pushed to the meta-information primitive. When encoding a new block into the meta-information TLV there are three requirements, a type number, length and value. The type number assignment is discussed in section 3.4.2. The NDN-CXX library comes with a block helper class for encoding different formats, such as string or integer values. An empty block can be used to signify if a piece of data is pushed. An empty block refers to the value section being empty but the type and length components still contain information. The existence of an empty block indicates that the data packet has been pushed. This was determined to be the most memory efficient implementation of the pushed member variable, keeping the header size to a minimum.

With the updated meta-information primitive, the Data packet can now be updated to include getter and setter methods that allow for conditional logic to be performed on a Data packet within the forwarding pipelines as outlined in section 3.3.

3.4.2 Type Number Assignment

A new TLV type is created for the pushed flag. NDN has a packet format specification [63] which defines how the TLV system is used in the NDN protocol and the type number reservations. A type number of 32 is reserved for pushed data. This was formerly reserved for *SelectedDelegation* but will have no impact or significance on the encoding and decoding of a data packet.

3.5 Scenario Design Considerations

This dissertation aims to evaluate the performance of the pure NDN, unsolicited data and proactive pushing data dissemination methods in a scenario where the data is transient and periodic. A GLOSA system was identified as a scenario that satisfies those criteria. The scenario development is broken down into two components, traffic simulation and network simulation. SUMO [56] is used to define the road topology and traffic demand, performing traffic simulations of an intersection for various speeds and densities of vehicle. NdnSim [8] is then used to define the network characteristics, varying the data update frequency and transmission range of nodes in the network.

It is possible to link the simulators, having ndnSim perform V2X communication during the runtime of a SUMO traffic simulation. To achieve this, a middleware such as VSimRTI can be used [64]. This allows for a full, comprehensive ITS simulation, where vehicles can alter their behaviour as a result of network communication between V2X applications. This approach is not necessary in the final design used to evaluate the performance of each data dissemination method. This dissertation aims to compare the performance of each pushing method under near identical network conditions. If nodes alter their behaviour during the course of a simulation due to the data dissemination method being used, then each comparison is not being subjected to exactly the same network conditions. The density of vehicles in a given area can alter as well as the speed vehicles are travelling. This is also the reason that the applications developed for use in the evaluation only emulate the communication patterns of a GLOSA system; it should not cause any dynamic behaviour in vehicles. The network conditions that each method of data dissemination is subjected to should be as similar as possible so as to have a fair and accurate comparison for transient and periodic data dissemination methods.

The approach implemented in this dissertation is that the path taken by each vehicle in each SUMO simulation is imported into ndnSim by the use of trace files. The trace files are output at the end of a SUMO traffic simulation and are used to inform ndnSim about the number of nodes required and each nodes mobility for the duration of the network simulation. The use of trace files ensures that the network conditions are as similar as possible for each data dissemination method. The design and implementation of traffic modelling is discussed

in section 3.6 and the design and implementation of the network simulation is discussed in section 3.7.

3.6 Traffic Modelling

SUMO is used to define the road topology and simulate traffic models. SUMO provides a well documented and mature set of tools to accurately simulate traffic models. On top of this, SUMO provides support to export trace files that can be then be used in ndnSim to define the nodes path during network simulations. There are two key parameters that will be defined in SUMO, speed and PCPH. Speed is configured when creating the road topology, as will be shown in section 3.6.1, and PCPH is configured when defining traffic flows, as discussed in section 3.6.3.

3.6.1 Road Topology

It is possible to quickly create a road topology using the OSM Web Wizard tool [65] that is packaged with SUMO. The tool opens a webbrowser and allows a user to select geographic regions from OpenStreetMaps. It also provides some controls for specifying random traffic demand for different traffic modes. Figure 3.6 shows an image of the OSMWebWizard portal.

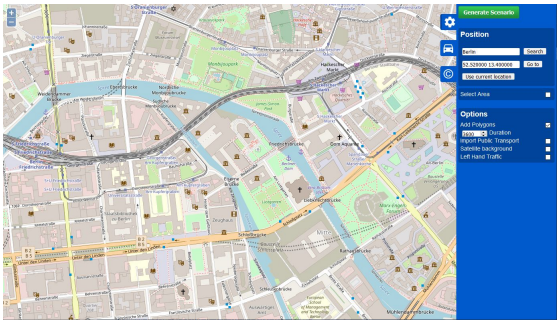

Figure 3.6: Web portal of OSMWebWizard

The tool is easy to use and allows users to quickly generate road topologies and traffic demand. OSMWebWizzard is not used in the final design because the tool lacks fine grained control over traffic demand, which is randomly generated when using the OSMWebWizard tool [56]. When attempting to re-create real world road topologies, the OSMWebWizzard tool would be an ideal candidate. But the scenario required for this dissertation is a simple intersection topology and so it is preferred to manually configure the road topology in order to have greater control over traffic demand in the simulations.

Manual configuration of road topology is achieved through the use of XML definitions that are input into NETCONVERT. NETCONVERT generates the road topology used by SUMO in simulating traffic. Road topologies are defined as a series of nodes and edges. Nodes are

defined in a .nod.xml file. Each row in the document defines a node, giving the node an id, x position, y position and optional type. This is seen below in figure 3.7.

```
<node id="1" x="-500.0" y="0.0" type="priority"/>
```

Figure 3.7: Node definition in .nod.xml

Edges are defined in a .edg.xml file. Edges define the connections between two nodes in the road topology. Each row in the schema defines an edge. An edge consists of an edge ID, from node ID, to node ID, lane priority, the number of lanes and the max speed of a lane. The *priority* is an ordinal number that determines right-of-way rules. An edge definition can be seen velow in figure 3.8. The edges and nodes are all that NETCONVERT requires to generate the topology of an intersection.

```
<edge id="2i" from="2" to="0" priority="78" numLanes="3" speed="100" />
```

Figure 3.8: Edge definition in .edg.xml

The incoming and outgoing edges to an intersection must also be defined. This is done in a .con.xml file. Vehicles in the network that are approaching an intersection need to know whether they may perform a right turn, left turn, U turn or continue straight and which lane a vehicle needs to be in to perform these actions.

Combining the .con.xml, .nod.xml and .edg.xml file, the four way intersection shown in figure 3.9. Each road consists of six lanes, three in and three out. NETCONVERTs heuristic refinement of missing data means that not every detail needs to be defined [56], allowing unimportant features to be completed by NETCONVERT. Certain features of the road topology, such as right of way rules at the intersection, are not of major concern so long as each lane receives the intended PCPH.

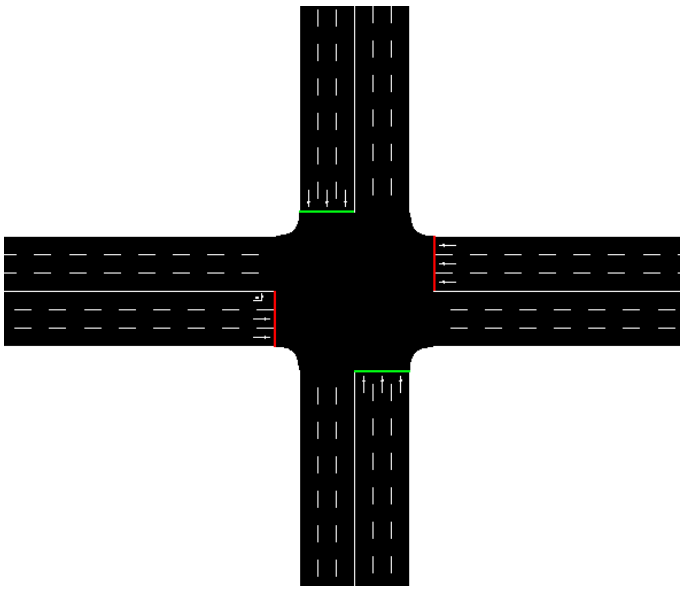

Figure 3.9: Intersection created by NETCONVERT

3.6.2 Traffic Light System

As explained in section 1.2.5, a fully functional GLOSA system is not implemented. This means that a static TLS can be defined using SUMO. A static TLS refers to the sequence of signal phase and timings being fixed. NETCONVERT will automatically generate traffic light programs for junctions during the construction of networks.

3.6.3 Traffic Demand

Traffic demand for simulations in SUMO can be defined as singular trips, routes or flows as mentioned in section 2.7.2. The definition of traffic demand is done manually using XML and is defined in a .rou.xml file. A row can define one of four things, a vehicle type (vType), a route, a vehicle, or a flow. A vehicle type defines the characteristics of a type of vehicle in the network, such as acceleration, deceleration, top speed, length and so on. A vehicle is an instance of a vehicle type, with an ID, a departure time, a colour, and a route. A route defines a series of edges in the network to be traversed. A flow defines repeated vehicle emissions into the network following a distribution or equi-distant time spacing.

Flows are used for the generation of vehicles in the simulation. With flows, vehicles will enter the network distributed either equally or randomly over a given time period, depending on the chosen flow and value. There are three possible types of flow to choose from, *vehsPerHour*, *period*, and *probability*. In the *vehsPerHour* flow, vehicles enter the network equally spaced, determined by the chosen value, to ensure that an exact amount of vehicles travel on a lane per hour. If a value of two hundred is chosen, then a vehicle will enter the network every eighteen seconds, per lane. The *period* flow defines the equally spaced insertion of vehicles into the network at a given period. The *probability* flow has a chance of inserting a vehicle into the network each second with probability p.

The vehicles per hour flow was chosen to satisfy the PCPH parameter requirement, discussed in section 3.1.1. Each lane has a flow specifying the flow ID, start time, end time, from lane, to lane and the vehPerHour, as seen in figure 3.10.

```
<flow id="f11" begin="0" end="10000" from="1i" to="1o" vehsPerHour="1900"/>
```

Figure 3.10: Flow definition in .rou.xml

After defining the traffic demand for a simulation the road topology will be populated with vehicles when a simulation is run. Vehicles will be inserted into the road network following the flow that has been defined. Figure 3.11 shows the intersection populated with vehicles after the traffic demand has been specified.

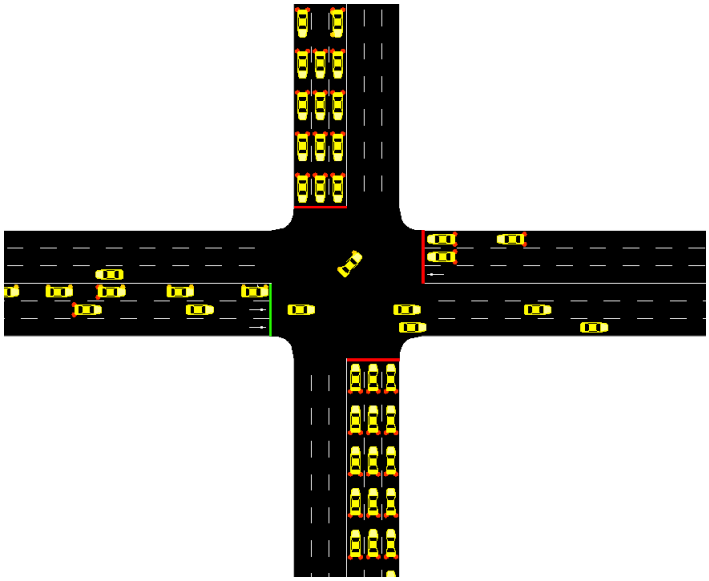

Figure 3.11: Intersection created by NETCOVERT with traffic demand

3.7 Network Scenario

The network scenario design can be broken down into three components, the consumer application, producer application and scenario configuration. A custom consumer and proactive producer application are developed. These custom applications as well as the ndnSim example producer application are used to exhibit a communication pattern similar to that of a GLOSA system. Network configuration and setup is performed in what ndnSim calls a scenario. The network scenario defines the nodes in the network, the mobility patterns that they will follow, the full network stack running on the nodes and the applications installed on each node. The network scenario is also where the run-time for the simulation is defined as well as defining the network tracing to be used over the duration of a simulation, which will provide the data required to evaluate each simulation.

3.7.1 Applications

Applications in ndnSim inherit from the *ndn::App* class which contain methods triggered when packets arrive at or leave the application layer through an application interface. To receive packets the *OnData*, *OnInterest*, and *OnNack* methods of the *ndn::App* class need to be overridden. To send packets to the instance of the NDN protocol on a node, the *SendData* and *SendInterest* methods of the *ndn:App* class should be used, passing a correctly formed Data and Interest packet respectively.

3.7.2 Consumer Application

A custom consumer application is designed and implemented that periodically requests updates for content under a given namespace. The consumer application has three configurable attributes, the name of the data to be requested, the frequency that data will be requested, and a trace object for recording application level metrics. Upon startup, the application waits a random back-off time before requesting the first packet. This time is determined by a uniform distribution between 0 and $2 * 1/frequency$ which is seeded to ensure reproducibility. This is done to ensure that all consumers in the network do not request updates at the same time as this is unlikely to occur in a real world scenario. After requesting the first packet, a consumer application requests updates at a fixed period. Request scheduling is performed by the NS-3 scheduler. NS-3 [54], which is the basis that ndnSim is built from, uses its own internal simulation time. When scheduling events during the course of a simulation, the NS-3 scheduler should be used to ensure that events are scheduled with respect to the simulation time and not the system time.

Algorithm 3.3: Push VNDN Event loop for Consumer c

Possible Events: startup s, creation of interest *i*, arrival of data *d*

```
1  switch event do
2      case s do
3          Schedule Next Packet;
4      case i do
5          create valid interest i;
6          tell i it must collect fresh data;
7          broadcast interest i;
8          Schedule Next Packet;
9      case d do
10         Send information about arrival of packet d to application layer tracing;
11 end
```

3.7.3 Producer Applications

Two producer applications are used. For simulations using the pure NDN protocol and for unsolicited data scenarios, the example producer application that is packaged with ndnSim is used. For simulations using the proactive pushing method, a custom producer application is used that is able to push data packets into the network. For all producer applications, the freshness period of a piece of data is the same as the update frequency for the transient data. The size of the data packet is fixed at 600 bytes. This was found to be the upper bound of CAM packet sizes of a Renault vehicle in an urban or highway environment [66].

Pure NDN and Unsolicited Data Application

In the pure NDN and unsolicited data scenarios the producer is required to reply to arriving Interest packets with Data packets of a set size and with an identical name to the incoming interest. The example producer application satisfies these requirements. It has a set of attributes that can be set defining the namespace it contains data for, the payload size of data packet responses, the freshness period of data packets it creates, application specific signatures for security features and a postfix that is appended to the name of created data packets. Only the namespace, payload size and freshness period attributes need to be configured for a scenario evaluating pure NDN or unsolicited data.

Proactive Pushing Application

The proactive producer application has two functions. It will respond with a data packet to any for data in a given namespace and it will push data packets into the network at a fixed period under the same namespace. The proactive producer application has four attributes, a name prefix that the application responds to, the payload size for data packets it produces, the freshness period of data packets it produces, and the update frequency for the transient data that the proactive producer is pushing into the network. As with the consumer application, the proactive producer will wait a random back-off time defined by a uniform distribution before starting to push data into the network at a given period.

The proactive producer requires an entry in its FIB for the namespace under which it pushes data. To create an entry, the "Face" of a network device will be required as well as an assigned cost. All communication in NDN is performed through the "Face" abstraction, regardless of whether the packet originates from an application on the node or from a network device. The NDN protocol API, installed on each node, is used to get the Face associated with a given network device. The NDN API is then used once again to add the network device Face, with a cost of 1, as a path for data in the nodes FIB. A cost of one is provided so that the Forwarding Strategy of a node uses the manually configured path.

There are two scenarios when a producer will forward data into the network, when responding to an interest packet or when pushing data into the network. The only difference between the state of the two generated data packets is that one is pushed and the other is not. A boolean flag is added to the application logic for sending data packets to indicate whether the generated packet should contain the pushed TLV block, discussed in section 3.4, or not.

Algorithm 3.4: Push VNDN Event loop for Producer p

Possible Events: startup s, arrival of interest *i*, pushing of *d*

```
switch event do
    case s do
        Get Network Interface;
        Add path to FIB for prefix with interface and cost 0;
        Schedule Next Packet;
    case i do
        SendData(false);
    case d do
        SendData(true);
        Schedule Next Packet;
end
Function SendData(isPushed: bool) is
    Create valid data packet;
    Set freshness period to request frequency;
    if isPushed = true then
        set pushed flag of data packet to true value;
    end
    Send Data packet to NDN protocol;
end
```

3.7.4 Scenario Design

To create a scenario in ndnSim, the environment in which simulations will run must be configured. As mentioned in section 2.7.1, ndnSim has helper classes that can be used to configure the network stacks of each node in the simulation, install applications, configure tracing, configure how nodes move in the network and much more.

The components that need to be configured for the simulation are the mobility models of nodes, the physical layer characteristics, the data link layer characteristics, the network/transport layer, the applications running on the nodes, and tracing to gather data for evaluation.

NdnSim inherits mobility models from the NS-3 simulator. These models allow nodes to physically move over the course of a simulation. The models are simplistic, allowing for nodes to perform random walks, move in straight lines or follow some pre-defined route. It quickly becomes apparent that the pre-defined mobility models are not sufficient when trying to realistically simulate traffic flow. This is why the trace file function from NS-3 is used. The trace file explicitly defines the route of every node in the simulation and is coded in an NS-3

specific format. SUMO can export the route taken by every vehicle in a traffic simulation as an NS-3 formatted trace file. The trace file method is used to define the mobility of the consumer nodes during a ndnSim simulation. The producer node, which is a single TLS at the centre of an intersection, uses a constant position mobility model that means it is fixed in place.

The IEEE WAVE protocol [41] is chosen for the physical and data link layer over a cellular equivalent such as LTE-V [67]. The WAVE protocol is well supported in NS-3, allowing for the simple and accurate physical and data-link layer implementation for a VANET scenario. The Yet Another Network Simulator (YANS) [68] helper within NS-3 is used to configure the WAVE network controllers on each node. NS-3 allows the stacking of different propagation delay and loss models for tuning wireless channel characteristics through various combinations of pre-built and custom models. The transmission range of a node can be configured using YANS. This is achieved through using YANS implementation of the constant speed propagation delay model and YANS log distance propagation loss model, varying the exponent value of the propagation loss model to control transmission range.

A propagation loss model defines how signal power decreases as it travels through the physical medium. The log distance model is the default model for YANS calculated as such $L = L_0 + 10n \log(\frac{d}{d_0})$ where L_0 is the path loss at the reference distance, n is the path loss distance exponent, d is the distance in metres, and d_0 is a reference distance. In the case of YANS, the reference distance d_0 is 1 metre and the default value of the exponent n is 3 [68].

A propagation delay model defines the speed at which a signal travels through the physical medium. Using a constant speed propagation delay model results in signals travelling at a fixed speed which is the speed of light in vacuum by default. Unfortunately, the only reasoning for this choice is the transmission range for nodes in the network could not be altered unless this model was used with the log loss model as described above. The use of a constant speed propagation model will not harm the results of the simulation as the differences in performance of each data dissemination mechanism will be the same.

Through experimentation it was found that setting the exponent of the log loss model to 3, 2.72 and 2.55 will provide a transmission range of 100m, 200m and 300m respectively.

The NDN protocol is used for the network/transport layer of each node. NdnSim has the *ndn::StackHelper* class to allow for the quick and easy installation and configuration of the NDN protocol on each node in the network. Using the stack helper, the consumer nodes are configured with default routes, are set with a CS size of 2, though this is not required, and are set to use a least recently used cache replacement policy. The only required difference for the producer node is disabling the cache. The producers cache is disabled to prevent the constant caching of data packets that the node is producing as it was found to inhibit the dissemination of pushed data. For a production system the cache needs to be enabled so the

TLS could take on the role of a road-side unit (RSU).

The final scenario configuration step is installing applications on nodes in the network. Three applications are used, one consumer application for all simulations and two producer applications, one for base NDN and unsolicited data dissemination simulations and one for proactive pushing simulations. The *ndn::AppHelper* is used to install applications on nodes in the network. This helper class allows the application attributes to be set.

After configuration the scenario is ready to run. NdnSim uses the python build tool waf to compile and run scenarios. Figure 3.12 shows a running simulation being visualized using the pyviz application that comes with ndnSim.

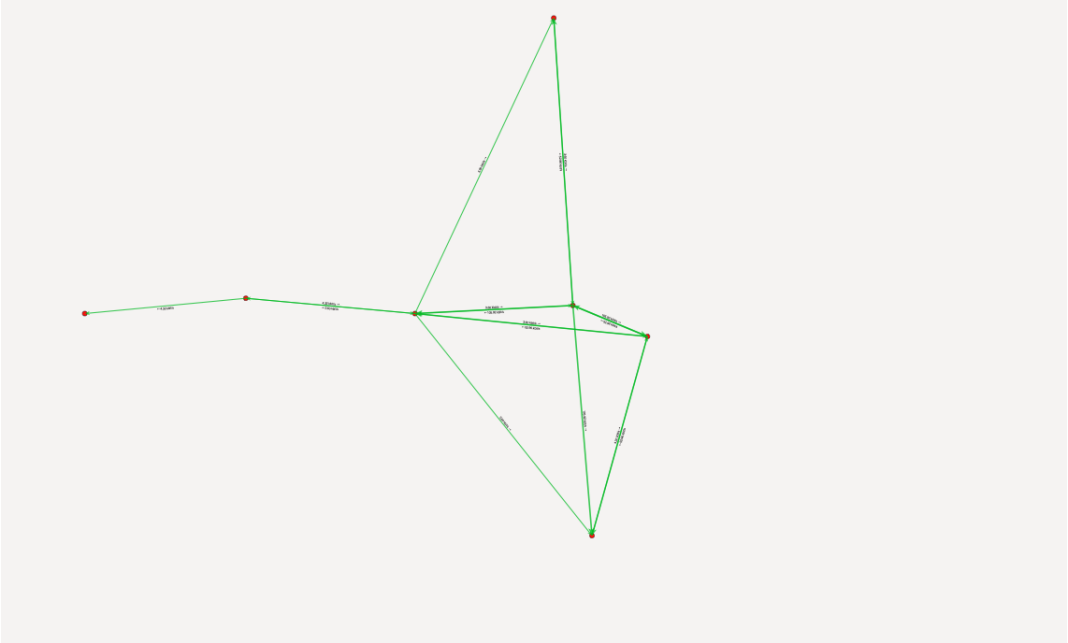

Figure 3.12: Running simulation using base NDN, visualized using pyviz

3.7.5 Summary

This chapter detailed the design and implementation of two additional data dissemination methods into the NDN protocol, unsolicited data and proactive pushing. The design and implementation of a scenario used to evaluate each method and pure NDN was also discussed. Both unsolicited data and proactive pushing were successfully implemented, but the forwarding of data in the proactive pushing method could not be achieved during the time frame of this project. A scenario, consisting of a four way intersection with a TLS at its centre, is implemented using SUMO and ndnSim. SUMO is used to perform the traffic modelling which is recorded in trace files. NdnSim is used to perform network simulations, importing the trace files exported by SUMO to define the consumers mobility.

4 Evaluation

In this chapter, the performance of unsolicited data and proactive pushing are examined. Pure NDN is used as the control case. Evaluation is performed using the scenario described in section 3.7.4; a four lane intersection with a TLS exhibiting the communication pattern of a GLOSA system. A simulation is run for each combination of vehicle density, vehicle speed, update frequency and node transmission range, totalling 162 unique simulations. Congestion, delay and cache hit ratio, defined below in section 4.1.1, are the three metrics used to measure network performance.

4.1 Data Dissemination Method Testing

To ensure a fair and accurate evaluation of pure NDN, unsolicited data and proactive pushing, each method is evaluated using the same trace files. Each method is simulated in ndnSim for all combinations of trace file, transmission range and update frequency. All distributions are seeded to ensure deterministic behaviour where possible, such as the back-off time before a node starts to communicate. The possible values of each parameter, defined in section 3.1.1, are shown below.

- PCPH: 285v, 950v and 1900v where v stands for vehicles per lane per hour
- Vehicle Speed: 30km/h, 60km/h and 100km/h where km/h stands for kilometres per hour
- Transmission Range: 100m, 200m, 300m where m stands for metres
- Update Frequency: 100ms, 1s

The HCM manual [59] defines 1900 PCPH as 100% utilisation of a lane. The values of 285v, 950v, and 1900v, representing 15%, 50% and 100% lane utilisation respectively, are chosen to represent sparse, medium and dense scenarios.

4.1.1 Testing Metrics

As stated in chapter 1, transient and periodic data in NDN could have a negative affect on the congestion and delay experienced in a network. Therefore, the metrics described below are chosen based on their suitability in measuring the performance of a network in terms of congestion and delay. The three metrics used are congestion, delay and cache hit ratio.

Congestion

Congestion is measured as the total number of packets in the network for all simulations. The rate tracer provided by ndnSim is used to calculate the congestion in the network. Any packet being received or sent from a face in a node is recorded; this includes application and system-level faces. The rate tracer encodes a timestamp, node ID, face ID, face description, packet type, packet rate, kilobyte rate, packet numbers and kilobyte throughput into each recorded metric. The only desired information is the timestamp, face description and packet count. The packet numbers is the total number of packets recorded during a time period, which is one second for the simulations. The face description is required to filter out any non-network face.

As the number of packets in the network increases, the network overhead increases, which increases the likelihood of negatives effects such as packet drop and low throughput [36].

Delay

Delay is measured as the average time taken for the most recent interest to be satisfied for all nodes in the network over all simulations. The application tracer provided by ndnSim is used to calculate delay. Delay is measured as last delay, which is the time taken for the most recently created interest for a piece of data to be satisfied. The application tracer encodes a timestamp, node ID, application ID, sequence number, delay type, delay in seconds, delay in micro-seconds, a re-transmission count and a hop count into each recorded metric. For the purpose of this dissertation only the timestamp, delay type, delay in seconds and hop count are used. The delay type can be either *last delay* or *first delay*. First delay is the time taken for the first interest created for a piece of content to be satisfied. Last Delay is the time taken for the most recently created interest for a piece of content to be satisfied.

Cache Hit Ratio

Cache Hit ratio is measured as the average cache hit ratio for all nodes for all simulations. A cache hit ratio is the number of cache hits over the number of cache hits and cache misses. The CS tracer provided by ndnSim is used to calculate the cache hit ratio of each node during a simulation. In-network caching is an important feature of the NDN protocol and is an effective tool for disseminating information. A cache hit ratio can indicate how effectively data is being

disseminated in the network. The CS tracer encodes into each metric a timestamp, node ID, type of counter for the time period and packet count for the time period. The type of counter for the time period can be one of *cache hits* or *cache misses*. For better processing the metrics are converted to a tidy data format [69] where cache hits and cache misses are their own variables. The cache hit ratio is calculated with the formula described below.

$$Ratio = \frac{CacheHits}{CacheHits + CacheMisses}$$

4.2 Results

162 unique simulations were run, each simulation ran for a duration of 100 seconds and every combination of vehicle density, vehicle speed, transmission range, update frequency were tested for each method. All vehicles in the network drove using the same model implemented in SUMO and developed by Krauß[58]. Network conditions were identical in each simulation apart from the data dissemination method used. The physical and data link layers use the WAVE protocols. The link speed is 6Mbps with a channel bandwidth of 10MHz. Vehicles entered the network at the incoming links on the edge of the road topology, labelled 1, 2, 3 and 4 in figure 4.1. The comprehensive testing done allowed each method to be evaluated in a number of ways. It was possible to see how the performance of each method improved or degraded as either vehicle density, vehicle speed, update frequency or node transmission range changed. All graphs show the performance of each method for each update frequency.

Attribute	Value
Phy/Mac layer	IEEE 802.11p, IEEE 1609.4
Bandwidth	10MHz
Bit Rate	6Mbps
Network/Transport layer	NDN
Applications	Consumer, Standard Producer, Proactive Producer
Simulation Time	100 seconds
Data Packet Size	600 bytes
Vehicle Density	15%, 50%, 100%
Vehicle Speed	30km/h, 60km/h, 100km/h
Transmission Range	100m, 200m, 300m
Update Frequency	1s, 100ms

Table 4.1: Table showing the configurable attributes and values used in the scenario

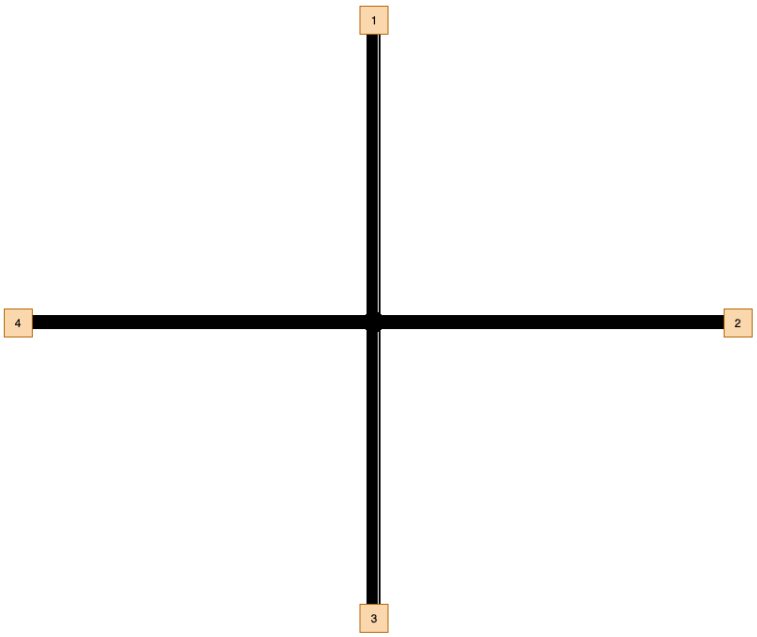

Figure 4.1: Test road topology

4.2.1 Congestion

The results for congestion can be seen in figure 4.2. Unsolicited data and proactive pushing resulted in a decrease in congestion of 75% and 67.2% respectively, at a frequency of 1s. There was a noticeable increase in overall congestion when the update frequency was increased to 100ms. This is expected with a reduced freshness period that requires all nodes in the network to request content more frequently. Proactive pushing resulted in more congestion in the network when compared to pure NDN (6.4%) at a update frequency of 100ms. In general, when compared to pure NDN, unsolicited data produces the least amount of congestion, with a reduction of 75% and 21.4% at update frequencies of 1s and 100ms respectively.

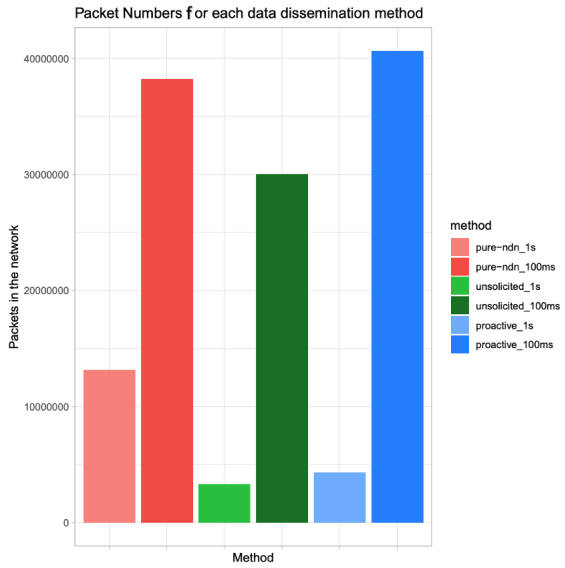

Figure 4.2: Overall results for congestion from all simulations

The behaviour of each data dissemination method as the density of vehicle changes can be seen in figure 4.3. At a update frequency of one second, congestion in the network increases as the density of vehicle increases. Both unsolicited data and proactive pushing resulted in far fewer packets in the network than the control case of pure NDN. Unsolicited data is marginally better than proactive pushing. At a update frequency of 100ms, as the density of vehicle in the network increased, the use of the proactive pushing resulted in a much greater number of network packets. In this case, the proactive pushing method causes more congestion caused than pure NDN.

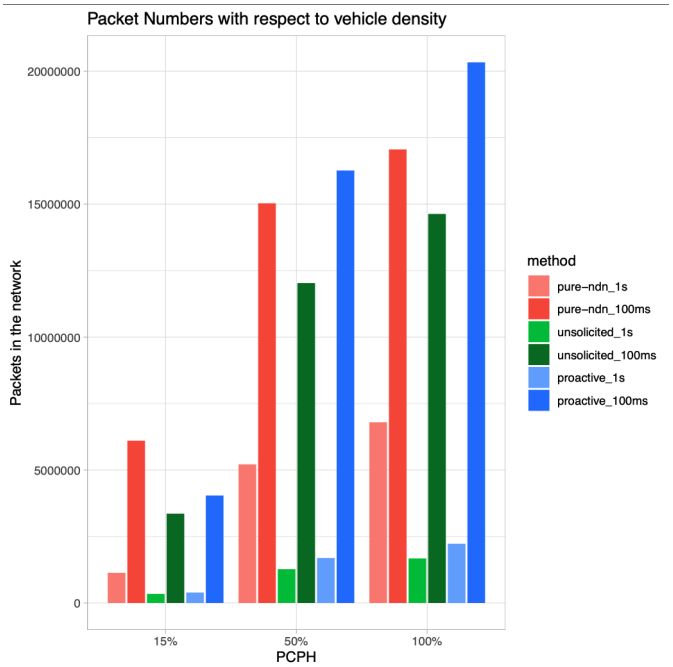

Figure 4.3: Graph showing results for congestion, grouped by the density of vehicle

The behaviour of each data dissemination method as transmission range changes can be seen in figure 4.4. As transmission range increased, congestion in the network was expected to diminish; this does not appear to be the case. At an update frequency of 1s, the pure NDN method increases congestion in the network as transmission range increases, whereas unsolicited data and proactive pushing remain relatively stable. At 100ms, the number of packets in the network for each dissemination method does not appear to be consistent. Pure NDN decreases as the transmission range increases, unsolicited data remains the same, and proactive pushing does not follow any discernible pattern.

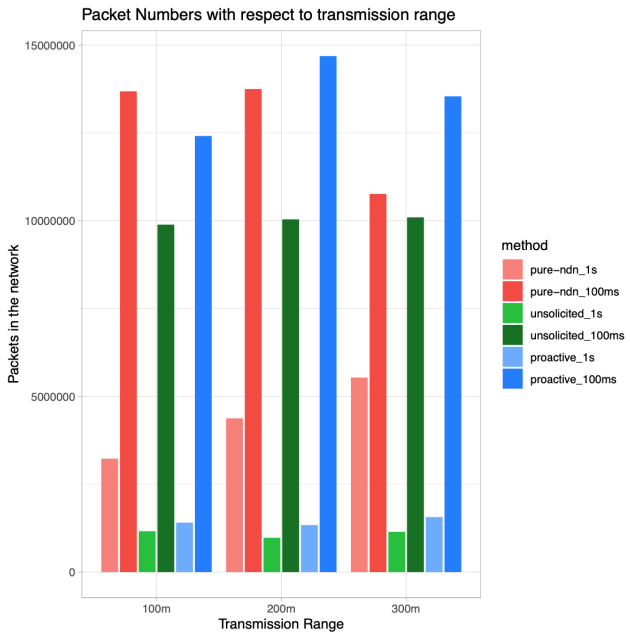

Figure 4.4: Graph showing results for congestion, grouped by transmission range

The behaviour of each data dissemination method as vehicle speed changes can be seen in figure 4.5. Changes in vehicle speed do not have an effect on congestion in the network of any method. This is understandable in the context of an intersection. Vehicles must slow as they approach an intersection potentially leading to similar communication times.

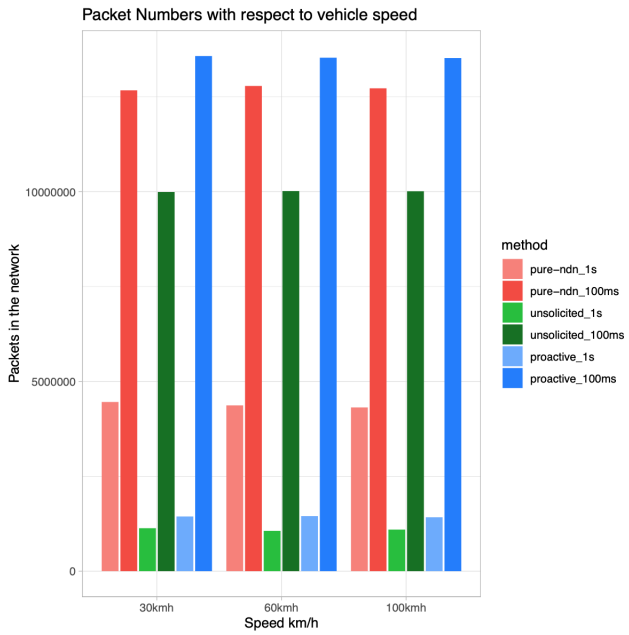

Figure 4.5: Graph showing results for congestion, grouped by vehicle speed

The distribution of the congestion measurements in one second time periods for each method, seen in figure 4.6, displays a positive correlation between update frequency and the variance of packet numbers in the network. There is a large variance in congestion at an update frequency of 100ms.

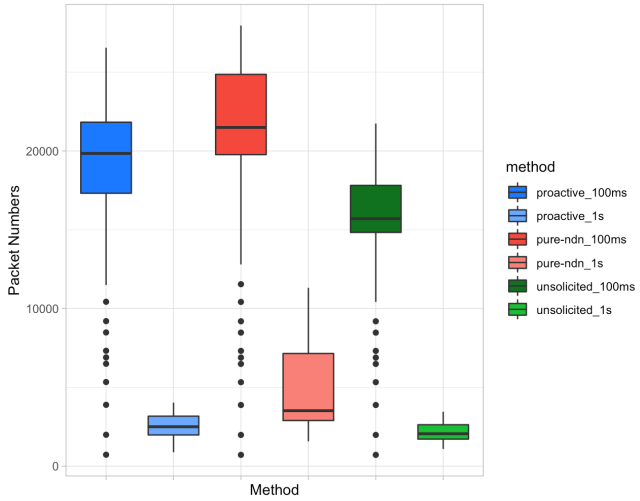

Figure 4.6: Boxplot for the number of packets in the network per method

4.2.2 Delay

The results can be seen in figure 4.7. The results showed that a pushing method greatly reduced delay experienced in the network. Unsolicited data and proactive pushing reduced delay by 73.2% and 59.7% at a update frequency of 1s, when compared to NDN. Unsolicited

data and proactive pushing reduced delay by 52.6% and 75.9% at a update frequency of 100ms, when compared to pure NDN. At a update frequency of 1s, the unsolicited data method outperforms the proactive pushing method, but when the update frequency is 100ms the proactive pushing method produces better results. It is observed that as the frequency of pushing increases, the mean delay for the proactive pushing method decreases.

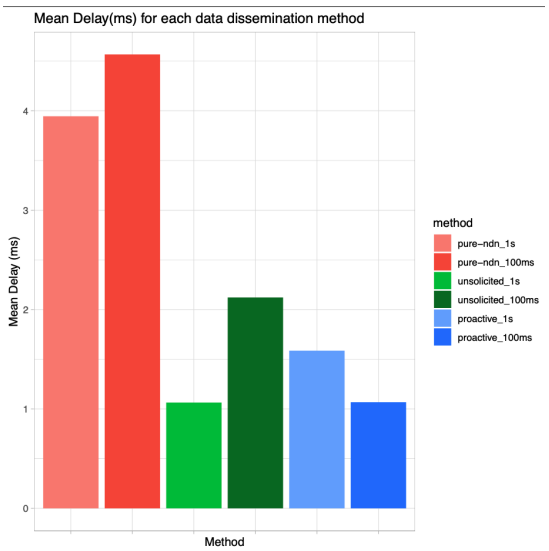

Figure 4.7: Overall delay of each method for all simulations

The behaviour of each data dissemination method as the density of vehicle changes can be seen in figure 4.8. At an update frequency of 1s, the mean delay of each method trends downwards as the density of vehicle increases, but at 100ms the opposite effect is observed. At a update frequency of 100ms there could be a greater chance for a node to not contain desired data in its cache, either because it is yet to receive a copy of the data or because its copy has become stale. As the density of vehicle increases, the number of nodes that require new data would be amplified. An update frequency of 1s may be a sufficiently long period of time to see greater impact of the positive effects on delay of pushing data.

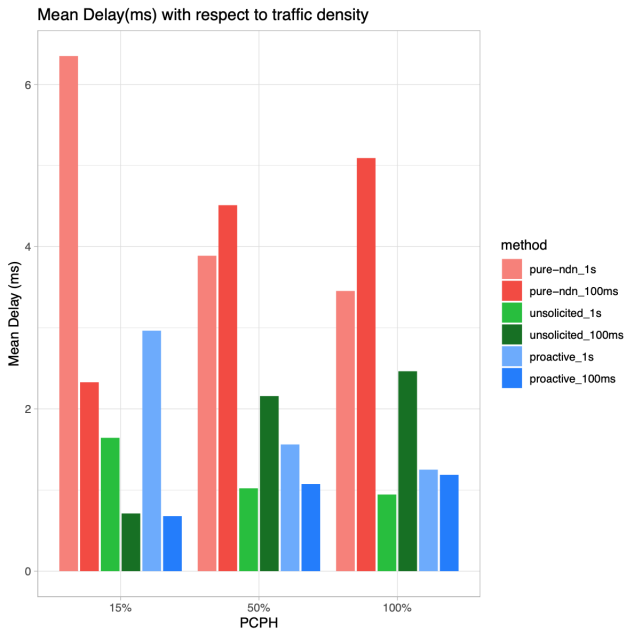

Figure 4.8: Graph showing results for delay, grouped by the density of vehicle

The behaviour of each data dissemination method as transmission range changes can be seen in figure 4.8. At an update frequency of 1s, there is a downwards trends for the delay of each method as the transmission range of nodes increases. The unsolicited data method performs the best in this case. It was expected that the proactive pushing method, at a higher update frequency, would perform the best as the transmission range increased. While this is the case for ranges of 100m and 200m, at 300m the delay actually increases. This could possibly be an artifact of the scenario setup. The edge of the simulation is at 400m and every node that is yet to start following its route waits at the edge of the simulation. It is possible that at a transmission range of 300m nodes that are waiting at the simulation edges are communicating with nodes in the process of following their pre-defined route. This communication with nodes that are not yet fully participating in the simulation would increase mean delay.

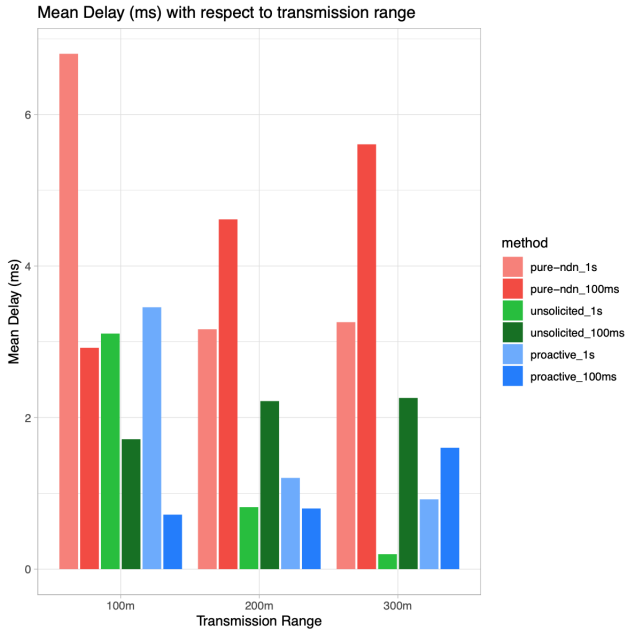

Figure 4.9: Graph showing results for delay, grouped by transmission range

Delay experienced in the network as the speed of vehicles changes remains steady. Once again this is to be expected as vehicles slow down when approaching the intersection.

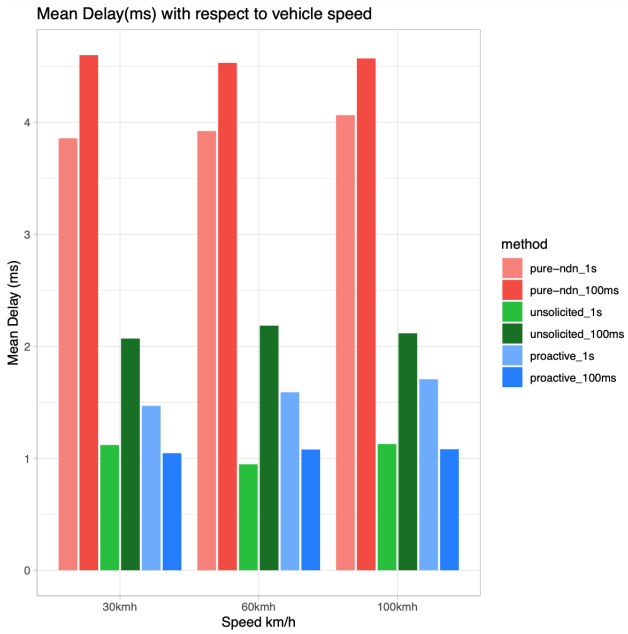

Figure 4.10: Graph showing results for delay, grouped by vehicle speed

Due to the scenario design, there is always a large spike in delay as the first vehicles come within range of the producer. After the initial spike, when a stream of vehicles are entering communication range, the delay in the network drops significantly. This can be seen in figure 4.11. To account for this period before the network reaches a *settled* state, only data 20 seconds after the simulation starts is included when calculating the results.

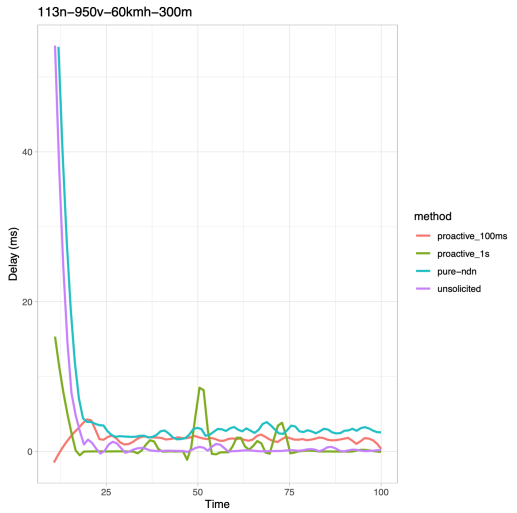

Figure 4.11: Line chart using loess regression for a single simulation showing delay for each method over time

4.2.3 Cache Hit ratio

The effective use of in-network caching was measured using the cache hit ratio of all nodes in the network. The results can be seen in figure 4.12. The results showed that proactive pushing has the highest cache hit ratio at 31.38% and 47.17% at an update frequencies of 1s and 100ms respectively. This is in line with expectations as the proactive pushing method *places* data in the CS of nodes in the network, before they require the data. Unsolicited data at 100ms is not in line with expectations, with a cache hit ratio of 30%, which is less than the 36.4% cache hit ratio of pure NDN.

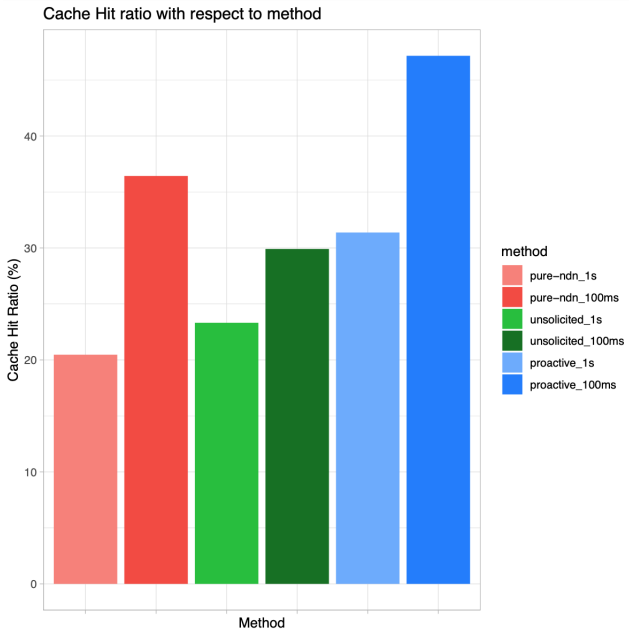

Figure 4.12: Overall results for cache hit ratio for all simulations

The behaviour of each data dissemination method as the density of vehicle changes can be seen in figure 4.13. The cache hit ratio for each method has a positive correlation with the density of vehicle in the network. The proactive pushing method at a 100ms update frequency performs the best and this is in line with expectations. As stated earlier, the unsolicited data method at a 100ms update frequency under-performs.

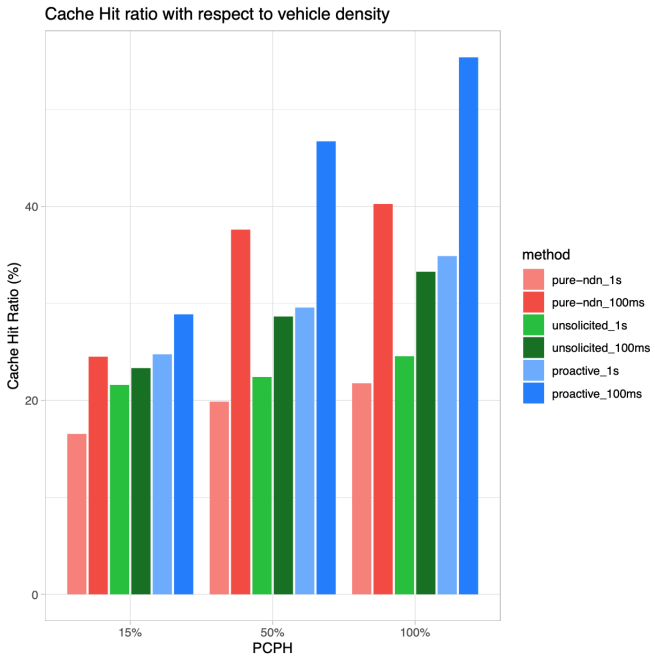

Figure 4.13: Graph showing results for cache hit ratio, grouped by the density of vehicle

The behaviour of each data dissemination method as transmission range changes can be seen in figure 4.14. The proactive pushing method achieves a greater than 80% cache hit ratio at a transmission range of 300m and update frequency of 100ms. There is a clear positive correlation between cache hit ratio and transmission range. This makes sense as the larger the communication range, the higher the number of nodes that will receive data directly from a producer. At a transmission range of 100m, both the unsolicited data and proactive pushing method have a lower cache hit ratio than pure NDN, which is unexpected. It could be possible that 100m is not a sufficient distance from the producer to make effective use of a nodes cache in a GLOSA scenario, as nodes move in and out of the radius of communication relatively quickly.

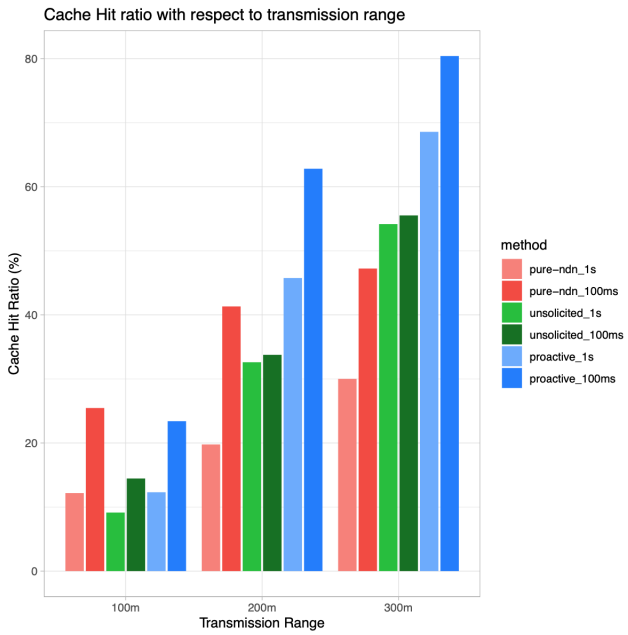

Figure 4.14: Graph showing results for cache hit ratio, grouped by transmission range

The behaviour of each data dissemination method as transmission range changes can be seen in figure 4.15. There is a slight downwards trend of the cache hit ratio of each method with respect to vehicle speed. Some nodes should pass through the intersection at their maximum speed during a green phase leading to a possibly reduced chance of receiving data into the cache of a node.

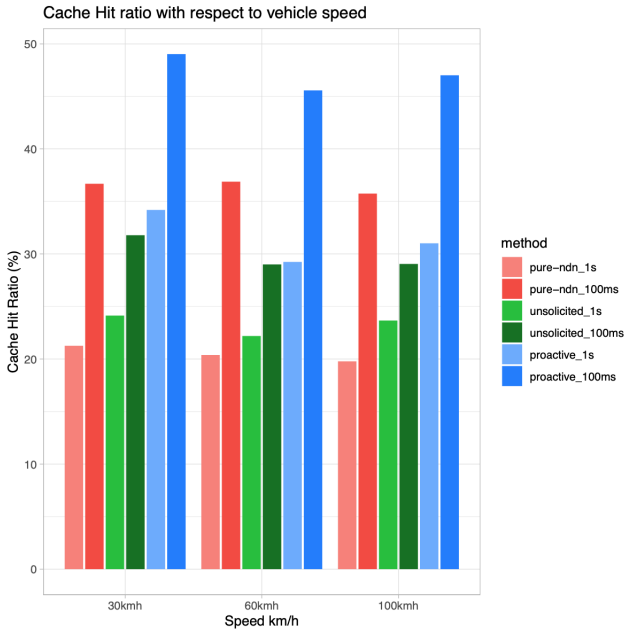

Figure 4.15: Graph showing results for cache hit ratio, grouped by vehicle speed

The cache hit ratios were generally below expectations. After testing the scenario setup it was determined that nodes waiting at the edge of the simulation were responsible for lowering the

values. NdnSim requires that all nodes in a simulation are created at the start of the simulation. This leads to a large pool of static nodes at the outer edges that are communicating. These non-participant nodes do not have access to the data and therefore skew the cache hit ratio towards a lower value. This can be seen below in figure 4.16.

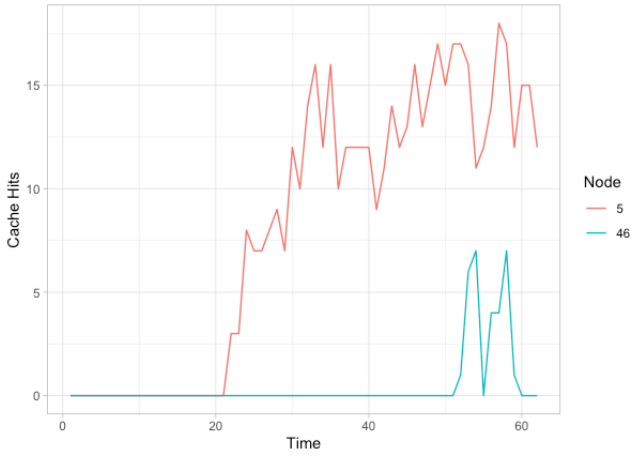

Figure 4.16: Example of the cache hit ratio of a node actively participating in a simulation (5) and a node which is static at the edge of the same simulation (46)

4.2.4 Summary

At a update frequency of 1s, unsolicited data and proactive pushing are an improvement over pure NDN in terms of congestion, delay and cache hit ratio. Proactive pushing has similar results to unsolicited data in terms of congestion and delay but is a marked improvement in terms of cache hit ratio. There appears to be a trade-off between increased network congestion and reduced network delay for the proactive pushing method at the higher frequency. This is seen when contrasting the results in figure 4.2 against figure 4.7, the delay results. In terms of congestion, the proactive pushing method performs noticeably worse than unsolicited data and is marginally worse than pure NDN. When looking at delay, proactive pushing outperforms both unsolicited data and pure NDN. The cache hit ratio of proactive pushing is a noticeable improvement over both pure NDN and unsolicited data. If the network can handle the increased congestion from proactive pushing at an update frequency of 100ms then it is a clear improvement. At an update frequency of 1s, the results show that unsolicited data and proactive pushing could be an ideal candidate for data which is both transient and periodic.

5 Conclusion

This project implemented and evaluated push-based data dissemination methods methods in the NDN protocol for the dissemination of transient and periodic data in VANETs. The hypothesis was that push-based methods improve network performance in terms of congestion and delay. The push-based data dissemination methods were unsolicited data and proactive pushing. The scenario used to evaluate each methods performance is designed to replicate vehicles approaching an intersection with a GLOSA enabled TLS, where every node in the network is emulating the communication pattern of a GLOSA system.

The results discussed in Chapter 4 show two stories when comparing each method with respect to the update frequency of the transient data. At a low update frequency of 1s, the unsolicited data and proactive pushing methods significantly improve the network performance when compared to the control case of pure NDN. At a 1s update frequency, unsolicited data performs better than proactive pushing in terms of congestion and delay in the network, but proactive pushing has a noticeably better cache-hit ratio. When the update frequency is 100ms, the improvement from pushing methods for network congestion is less noticeable, or worse, in the case of proactive pushing. At 100ms, pushing methods greatly reduce the delay experienced in the network, with proactive pushing performing the best. There appears to be a trade-off between congestion and delay with the proactive pushing method at higher update frequencies. The cache-hit ratio of the proactive pushing method is a significant improvement over the control case of pure NDN. The unsolicited data method performed the worst which was not expected.

Overall, the following results were gathered for each pushing method when compared to the control case of pure NDN. At an update frequency of 1s, the unsolicited data method resulted in 75% less congestion, 73.2% less delay, and a 13% increase in cache-hit ratio. At an update frequency of 100ms, the unsolicited data method resulted in 21.4% less congestion, 52.6% less delay, and a 17.9% decrease in cache-hit ratio. At an update frequency of 1s, the proactive pushing method resulted in 67.2% less congestion, 59.7% less delay, and 53.3% increase in cache-hit ratio. At an update frequency of 100ms, the proactive pushing method resulted in a 6.4% increase in congestion, 75.9% decrease in delay and a 29.5% increase in cache-hit ratio. These results provide valuable evidence towards proving the hypothesis that implementing

pushing methods into the NDN protocol for the dissemination of transient and periodic data in VANETs greatly improves network performance.

5.1 Future Work

The results gathered in this project are promising but can be further improved upon. The poor cache-hit ratio of unsolicited data at 100ms is unexplained and suggests that further work can be done on the scenario design and evaluation. The static nodes at the edge of the scenario were shown to be unduly skewing results in section 4.2.3 and should be removed in any future iteration of the scenario design. Evaluation of delay in terms of first delay and the time taken for the first interest packet to be satisfied, may be more accurate. A more rigorous examination of unexpected patterns identified in the results should also take place. A statistical analysis of the differences between each data dissemination method with respect to changing parameters should take place. Overall, there is potential for a more rigorous evaluation.

The algorithm used to implement the proactive pushing method should be updated to allow the forwarding of pushed data packets. Two possible solutions to be investigated are detailed in section 3.3.1. The first possible solution is to make nodes aware of many network interfaces to forward the pushed data to as well as the application face. Another potential solution is to have separate control flows for incoming and outgoing data. This could be achieved by checking the type of face that the data has arrived from. If it is an application face, then the data is outgoing and should ignore the Data in CS check. If the data has arrived from a network face, than the data is incoming and should be subjected to the CS check. After altering the proactive pushing method to forward data packets, the analysis should be rerun to see how the results change.

The scenario is also very simplistic, consisting of a single four lane intersection. A more complex road topology which consists of multiple intersections could be evaluated, similar to the scenario evaluated by Tielert et al. [44]. This evaluation would indicate whether the benefits of pushing methods are diminished by multiple competing intersections.

Bibliography

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," *Commun. ACM*, vol. 55, no. 1, p. 117–124, Jan. 2012. [Online]. Available: https://doi-org.elib.tcd.ie/10.1145/2063176.2063204

[2] A. John, "Why use a cdn? here are 10 data-driven reasons," 2019. [Online]. Available: https://hackernoon.com/why-use-a-cdn-here-are-10-data-driven-reasons-ee0a02672988

[3] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 66–73, Jul. 2014.

[4] H. Khelifi, S. Luo, B. Nour, H. Moungla, Y. Faheem, R. Hussain, and A. Ksentini, "Named data networking in vehicular ad hoc networks: State-of-the-art and challenges," *IEEE Communications Surveys Tutorials*, vol. 22, no. 1, pp. 320–351, Firstquarter 2020.

[5] "Ieee guide for wireless access in vehicular environments (wave) - architecture," *IEEE Std 1609.0-2013*, pp. 1–78, March 2014.

[6] G. Grassi, D. Pesavento, L. Wang, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang, "Acm hotmobile 2013 poster: Vehicular inter-networking via named data," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 17, no. 3, p. 23–24, Nov. 2013. [Online]. Available: https://doi.org/10.1145/2542095.2542108

[7] M. F. Majeed, S. H. Ahmed, and M. N. Dailey, "Enabling push-based critical data forwarding in vehicular named data networks," *IEEE Communications Letters*, vol. 21, no. 4, pp. 873–876, April 2017.

[8] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2: An updated NDN simulator for NS-3," p. 8.

[9] N. Team, "Nfd developer's guide," *Dept. Comput. Sci.*

[10] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named data networking (ndn) project,"

Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC, vol. 157, p. 158, 2010.

[11] R. A. Rehman and B. Kim, "Lomcf: Forwarding and caching in named data networking based manets," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 9350–9364, Oct 2017.

[12] Z. Yan, S. Zeadally, and Y. Park, "A novel vehicular information network architecture based on named data networking (ndn)," *IEEE Internet of Things Journal*, vol. 1, no. 6, pp. 525–532, Dec 2014.

[13] R. Bodenheimer, D. Eckhoff, and R. German, "Glosa for adaptive traffic lights: Methods and evaluation," in *2015 7th International Workshop on Reliable Networks Design and Modeling (RNDM)*, 2015, pp. 320–328.

[14] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 1024–1049, Second 2014.

[15] J. B. Kenney, "Dedicated short-range communications (dsrc) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, July 2011.

[16] H. Zimmermann, "Osi reference model - the iso model of architecture for open systems interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, April 1980.

[17] J. Postel, "Rfc0791: Internet protocol," 1981.

[18] D. Wing, "Network address translation: Extending the internet address space," *IEEE Internet Computing*, vol. 14, no. 4, pp. 66–70, July 2010.

[19] V. Fuller, T. Li, J. Yu, and K. Varadhan, "Classless inter-domain routing (cidr): an address assignment and aggregation strategy," 1993.

[20] P. Mockapetris and K. J. Dunlap, "Development of the domain name system," in *Symposium Proceedings on Communications Architectures and Protocols*, ser. SIGCOMM '88. New York, NY, USA: Association for Computing Machinery, 1988, p. 123–133. [Online]. Available: https://doi.org/10.1145/52324.52338

[21] J. Moy *et al.*, "Ospf version 2," 1998.

[22] A. Vakali and G. Pallis, "Content delivery networks: status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, Nov 2003.

[23] Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022 white paper," Tech. Rep., 2017. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[24] Yang-hua Chu, S. G. Rao, S. Seshan, and Hui Zhang, "A case for end system multicast," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1456–1471, Oct 2002.

[25] B. Daya, "Network security: History, importance, and future," *University of Florida Department of Electrical and Computer Engineering*, vol. 4, 2013.

[26] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 181–192. [Online]. Available: https://doi.org/10.1145/1282380.1282402

[27] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, "Developing information networking further: From psirp to pursuit," in *International Conference on Broadband Communications, Networks and Systems*. Springer, 2010, pp. 1–13.

[28] Named-Data, "NDN Packet Format Specification — NDN Packet Format Specification 0.3 documentation." [Online]. Available: https://named-data.net/doc/NDN-packet-spec/current/index.html

[29] A. Vakali and G. Pallis, "Content delivery networks: status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, 2003.

[30] C. Ghasemi, H. Yousefi, K. G. Shin, and B. Zhang, "Poster abstract: Routing meets caching in named data networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2018, pp. 1–2.

[31] A. Afanasyev, J. A. Halderman, S. Ruoti, K. Seamons, Y. Yu, D. Zappala, and L. Zhang, "Content-based security for the web," in *Proceedings of the 2016 New Security Paradigms Workshop*, ser. NSPW '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 49–60. [Online]. Available: https://doi.org/10.1145/3011883.3011890

[32] Z. Zhang, Y. Yu, A. Afanasyev, and L. Zhang, "Ndn certificate management protocol (ndncert)," *NDN, Technical Report NDN-0050*, 2017.

[33] U. F. C. Commission *et al.*, "Standard specification for telecommunications and information exchange between roadside and vehicle systems-5 ghz band dedicated short range communications (dsrc) medium access control (mac) and physical layer (phy) specifications," *Washington, DC (September 2003)*, 2003.

[34] H. Hartenstein and L. P. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *IEEE Communications Magazine*, vol. 46, no. 6, pp. 164–171, June 2008.

[35] M. Laroui, A. Sellami, B. Nour, H. Moungla, H. Afifi, and S. B. Hacene, "Driving path stability in vanets," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec 2018, pp. 1–6.

[36] Z. Cao, K. Shi, Q. Song, and J. Wang, "Analysis of correlation between vehicle density and network congestion in vanets," in *2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*, July 2017, pp. 409–412.

[37] D. Kim, Y. Velasco, W. Wang, R. N. Uma, R. Hussain, and S. Lee, "A new comprehensive rsu installation strategy for cost-efficient vanet deployment," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 5, pp. 4200–4211, May 2017.

[38] X. Ma, J. Zhang, X. Yin, and K. S. Trivedi, "Design and analysis of a robust broadcast scheme for vanet safety-related services," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 1, pp. 46–61, Jan 2012.

[39] D. Jiang and L. Delgrossi, "Ieee 802.11p: Towards an international standard for wireless access in vehicular environments," in *VTC Spring 2008 - IEEE Vehicular Technology Conference*, May 2008, pp. 2036–2040.

[40] "Ieee standard for wireless access in vehicular environments (wave) – multi-channel operation," *IEEE Std 1609.4-2016 (Revision of IEEE Std 1609.4-2010)*, pp. 1–94, March 2016.

[41] J. Bu, G. Tan, N. Ding, M. Liu, and C. Son, "Implementation and evaluation of wave 1609.4/802.11p in ns-3," in *Proceedings of the 2014 Workshop on Ns-3*, ser. WNS3 '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi-org.elib.tcd.ie/10.1145/2630777.2630778

[42] D. Su, M. Zargari, P. Yue, S. Rabii, D. Weber, B. Kaczynski, S. Mehta, K. Singh, S. Mendis, and B. Wooley, "A 5 ghz cmos transceiver for ieee 802.11a wireless lan," in *2002 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.02CH37315)*, vol. 1, Feb 2002, pp. 92–449 vol.1.

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

[44] T. Tielert, M. Killat, H. Hartenstein, R. Luz, S. Hausberger, and T. Benz, "The impact of traffic-light-to-vehicle communication on fuel consumption and emissions," in *2010 Internet of Things (IOT)*, 2010, pp. 1–8.

[45] W. Zimdahl, "Guidelines and some developments for a new modular driver information system," in *34th IEEE Vehicular Technology Conference*, vol. 34, 1984, pp. 178–182.

[46] *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*, ETSI, 2014.

[47] M. Amadeo, C. Campolo, and A. Molinaro, "Internet of things via named data networking: The support of push traffic," in *2014 International Conference and Workshop on the Network of the Future (NOF)*, Dec 2014, pp. 1–5.

[48] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, "Named data networking for iot: An architectural perspective," in *2014 European Conference on Networks and Communications (EuCNC)*, June 2014, pp. 1–5.

[49] J. François, T. Cholez, and T. Engel, "Ccn traffic optimization for iot," in *2013 Fourth International Conference on the Network of the Future (NoF)*, Oct 2013, pp. 1–5.

[50] Z. Zhu, S. Wang, X. Yang, V. Jacobson, and L. Zhang, "Act: Audio conference tool over named data networking," in *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, ser. ICN '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 68–73. [Online]. Available: https://doi-org.elib.tcd.ie/10.1145/2018584.2018601

[51] C. Bian, Z. Zhu, A. Afanasyev, E. Uzun, and L. Zhang, "Deploying key management on ndn testbed," *UCLA, Peking University and PARC, Tech. Rep*, 2013.

[52] R. Chiocchetti, D. Rossi, and G. Rossini, "ccnsim: An highly scalable ccn simulator," in *2013 IEEE International Conference on Communications (ICC)*, 2013, pp. 2309–2314.

[53] C. Tschudin, C. Scherb *et al.*, "Ccn lite: Lightweight implementation of the content centric networking protocol," 2018.

[54] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.

[55] J. Thompson and J. Burke, "Ndn common client libraries," *Technical Report NDN-0024, Revision 1. NDN Project*, 2014.

[56] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: <https://elib.dlr.de/124092/

[57] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO – Simulation of Urban MObility," p. 6.

[58] S. Krauß, "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics," Ph.D. dissertation, 1998.

[59] N. R. C. (U.S.), Ed., *Highway capacity manual*. Washington, D.C: Transportation Research Board, National Research Council, 2000.

[60] J. Yin, T. ElBatt, G. Yeung, B. Ryu, S. Habermas, H. Krishnan, and T. Talty, "Performance evaluation of safety applications over dsrc vehicular ad hoc networks," in *Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks*, ser. VANET '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 1–9. [Online]. Available: https://doi.org/10.1145/1023875.1023877

[61] M. M. Alotaibi and H. T. Mouftah, "Data dissemination for heterogeneous transmission ranges in vanets," in *2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops)*, 2015, pp. 818–825.

[62] Z. Xia, P. Yu, and Y. Zhang, "Improving traffic information retrieval in vanet with ndn," in *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, 2018, pp. 100–106.

[63] "Name — NDN Packet Format Specification 0.3 documentation." [Online]. Available: https://named-data.net/doc/NDN-packet-spec/current/name.html

[64] D. Rieck, B. Schünemann, I. Radusch, and C. Meinel, "Efficient traffic simulator coupling in a distributed v2x simulation environment," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTools '10. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010. [Online]. Available: https://doi.org/10.4108/ICST.SIMUTOOLS2010.8640

[65] "Tutorials/OSMWebWizard - SUMO Documentation," library Catalog: sumo.dlr.de. [Online]. Available: https://sumo.dlr.de/docs/Tutorials/OSMWebWizard.html

[66] "Survey on cam statistics," Car 2 Car communication consortium, 2018. [Online]. Available: https://www.car-2-car.org/fileadmin/documents/General_Documents/C2CCC_TR_2052_Survey_on_CAM_statistics.pdf

[67] R. Molina-Masegosa and J. Gozalvez, "Lte-v for sidelink 5g v2x vehicular communications: A new 5g technology for short-range vehicle-to-everything communications," *IEEE Vehicular Technology Magazine*, vol. 12, no. 4, pp. 30–39, Dec 2017.

[68] M. Lacage and T. R. Henderson, "Yet another network simulator," in *Proceeding from the 2006 Workshop on Ns-2: The IP Network Simulator*, ser. WNS2 '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 12–es. [Online]. Available: https://doi.org/10.1145/1190455.1190467

[69] H. Wickham *et al.*, "Tidy data," *Journal of Statistical Software*, vol. 59, no. 10, pp. 1–23, 2014.

A1 Appendix

A1.1 Code repositories

Scenario design repository:

`https://github.com/ChrisLynch96/transient-periodic-information-dissemination-in-VNDN`

NFD alterations repository:

`https://github.com/ChrisLynch96/NFD`

NDN-CXX alterations repository:

`https://github.com/ChrisLynch96/ndn-cxx`