

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institute, Affiliated to VTU)

Bull Temple Road, Basavanagudi, Bengaluru - 560019



Presentation Report under Alternate Assessment Tool (AAT)

Smart Emergency Alert Broadcasting in Vehicular Ad Hoc Networks Using NS-3

Submitted in partial fulfilment of the requirements for the award of degree

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

By

NAME 1	USN 1
NAME 2	USN 2
NAME 3	USN 3
NAME 4	USN 4

Under the guidance of

Dr. Sowmya K S
Associate Professor

**Department of Computer Science and Engineering
2026-2027**



B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institute, Affiliated to VTU)
Bull Temple Road, Basavanagudi,
Bengaluru – 560019

Department of Information Science and Engineering

C E R T I F I C A T E

This is to certify that the Presentation Report under Alternate Assessment Tool entitled “*Smart Emergency Alert Broadcasting in Vehicular Ad Hoc Networks Using NS-3*” is a bona-fide work carried out by **name (usn), name (usn), name (usn), name (usn)** in partial fulfilment for the award of degree of Bachelor of Engineering in **Computer Science and Engineering** from **Visvesvaraya Technological University, Belagavi** during the year **2026-2027**. It is certified that all corrections/suggestions indicated for Internal Assessments have been incorporated in the report deposited in the departmental library. The Presentation Report has been approved as it satisfies the academic requirements in respect of AAT prescribed for the Bachelor of Engineering Degree.

Dr. Sowmya K S
Course Coordinator

Dr. Selva Kumar S
Professor and HOD

ACKNOWLEDGEMENT

We express our sincere gratitude to our guide, Dr. Sowmya K S, Associate Professor, Department of Computer Science and Engineering, B.M.S. College of Engineering, for her guidance and encouragement throughout this Alternate Assessment Tool (AAT) work on Network Programming.

We thank Dr. Selva Kumar S, Professor and Head of the Department, for providing the facilities and the academic environment that made this work possible.

We also thank the Principal and the management of B.M.S. College of Engineering for their support, and the faculty and staff of the department for their help.

Finally, we thank our families and friends for their constant support.

TABLE OF CONTENTS

1. Abstract.....	1
2. Objectives.....	2
3. Introduction.....	3
3.1 Network Programming and Vehicular Networks.....	3
3.2 Application X: Smart Emergency Alert Broadcasting.....	3
3.3 Importance of Sockets, UDP and Broadcast.....	4
3.4 Tools and Environment.....	4
4. Literature Review.....	5
4.1 The Broadcast Storm Problem.....	5
4.2 Distance-Based and Contention-Based Broadcast.....	5
4.3 Probabilistic Dissemination.....	6
4.4 IEEE 802.11p and Channel Effects.....	6
4.5 Surveys and Recent Work.....	7
5. Research Gap.....	9
6. Problem Statement.....	10
7. Methodology and Implementation.....	11
7.1 Proposed System Architecture.....	11
7.2 Network Communication Model.....	11
7.3 Mathematical Model.....	13
7.4 Proposed Algorithm.....	14
7.5 Socket Programming Implementation.....	17
8. Performance Evaluation.....	20
8.1 Experimental Setup.....	20
8.2 Performance Metrics.....	20
9. Output and Results.....	22
9.1 Program Output.....	22
9.2 Visual Replay of the Simulation.....	24
9.3 Comparative Analysis.....	25
9.4 Validation of the Proposed Approach.....	30
9.5 Observations.....	31
10. Conclusion.....	32
11. References.....	33

LIST OF FIGURES

Figure 7.1	System architecture of the Smart Emergency Alert Broadcasting	11
Figure 7.2	Highway scenario: crash location, radio range and region of interest.....	12
Figure 7.3	Flowchart of the Smart Relay receive and relay logic.....	16
Figure 7.4	Sequence diagram of one alert dissemination round (illustrative	17
Figure 7.5	AlertHeader format and encapsulation of one alert frame.....	18
Figure 8.1	Experimental workflow from the ns-3 program to graphs and the	20
Figure 9.1	Output of version 1: single-hop UDP broadcast to four parked vehicles.....	22
Figure 9.2	Output of the final program with Smart Relay (60 vehicles, seed 1, ...	23
Figure 9.3	Summary output of blind flooding on the same traffic (seed 1).....	23
Figure 9.4	Crash site 1.5 ms after the crash: flooding has made 27	24
Figure 9.5	The whole 2 km road 30 ms after the crash. Lines show which car	24
Figure 9.6	Packet delivery ratio against traffic density.....	26
Figure 9.7	Percentage of alerts that reached every vehicle on the section.....	26
Figure 9.8	Transmissions needed per alert (broadcast overhead).....	27
Figure 9.9	Share of receivers that re-broadcast the alert.....	27
Figure 9.10	Alert frames lost at receivers (collisions), logarithmic scale.....	28
Figure 9.11	Mean end-to-end delay against traffic density.....	28
Figure 9.12	95th-percentile end-to-end delay.....	29
Figure 9.13	Jitter of the end-to-end delay between consecutive alerts.....	29
Figure 9.14	Alert goodput (throughput of useful alert data delivered).....	30

LIST OF TABLES

Table 4.1	Summary of the literature reviewed.....	7
Table 7.1	Simulation parameters.....	12
Table 8.1	Performance metrics.....	20
Table 9.1	Mean results over 10 seeds (best value in bold).....	25

CHAPTER 1

ABSTRACT

When a vehicle meets with an accident on a highway, the vehicles approaching it need to know within a fraction of a second. Vehicular Ad Hoc Networks (VANETs) let cars exchange such warnings directly over IEEE 802.11p radios, but an alert must usually be relayed over several hops to reach every car. The simplest relaying rule, blind flooding, makes every receiver re-broadcast at once, which overloads the shared channel and causes collisions: the broadcast storm problem.

This work designs, implements and evaluates a Smart Emergency Alert Broadcasting application in the ns-3.45 network simulator. Every vehicle runs a UDP socket application that receives accident alerts sent to a subnet broadcast address and decides whether to relay them. We propose Smart Relay, an application-layer scheme in which receivers farther from the sender wait less before relaying, and a pending relay is cancelled when a vehicle further along the direction of travel has already relayed. Smart Relay is compared with blind flooding and probabilistic gossip on a 2 km four-lane highway with 20 to 120 moving vehicles, Nakagami fading and background beacon traffic, over 180 simulation runs.

Smart Relay delivered 96.0 % to 99.3 % of alerts at 60 to 120 vehicles, against 71.0 % to 93.9 % for flooding, while sending up to 67 % fewer packets and losing about 95 % fewer frames to collisions. Its mean delay of 5.5 to 10.8 ms stays far below the 100 ms needed for safety warnings.

CHAPTER 2

OBJECTIVES

The objectives of this Alternate Assessment were:

- To understand socket programming concepts in a wireless network by building a complete network application in the ns-3 discrete-event simulator.
- To implement an emergency alert application for vehicles using UDP datagram sockets, subnet broadcast addressing and a custom application-layer header carried in every packet.
- To simulate a realistic vehicular network: an IEEE 802.11p ad hoc radio, a fading channel, moving vehicles on a four-lane highway, a crash that blocks traffic, and background beacon load.
- To implement and compare three multi-hop alert relaying schemes at the application layer: blind flooding, probabilistic gossip, and the proposed distance-based Smart Relay with directional suppression.
- To measure and analyse network performance in terms of packet delivery ratio, end-to-end delay, jitter, throughput (goodput), broadcast overhead and collisions, across six traffic densities with ten random seeds each.
- To present the behaviour of each scheme visually through an animated replay generated from the simulator's own packet trace.

CHAPTER 3

INTRODUCTION

3.1 Network Programming and Vehicular Networks

Network programming is the writing of software that communicates over a network through the interfaces the operating system provides, most commonly the Berkeley socket API. A program creates a socket, binds it to a local address and port, and then sends and receives data with calls such as `sendto()` and `recvfrom()`. The transport protocol chosen for the socket decides how the data travels. TCP provides a reliable, ordered, connection-oriented byte stream with acknowledgements and retransmissions. UDP provides connectionless datagrams with no delivery guarantee but with very low overhead, and it is the only one of the two that can send a single datagram to many receivers at once through broadcast or multicast.

A Vehicular Ad Hoc Network (VANET) is a wireless network formed by moving vehicles that communicate directly with each other (vehicle-to-vehicle, V2V) and with roadside units, without depending on fixed infrastructure. Vehicles are equipped with IEEE 802.11p (DSRC) radios operating in the 5.9 GHz band, which allow a car to transmit to its neighbours without first joining or associating with a network [9]. VANETs are expected to support safety applications such as collision warnings, emergency vehicle warnings and hazard notifications [11].

3.2 Application X: Smart Emergency Alert Broadcasting

The application built in this work is an emergency alert system for vehicles. When a vehicle detects that it has crashed, it immediately broadcasts an ACCIDENT ALERT to the vehicles around it and keeps repeating the alert every second. Every vehicle that receives the alert warns its driver, so that approaching vehicles can slow down before they reach the crash site.

A single 802.11p transmission reaches only a few hundred metres, and on a real highway the cars that most need the warning may be a kilometre or more behind the crash. The alert therefore has to be relayed from car to car over several hops. How each car decides whether, and when, to relay the alert determines whether the warning reaches everyone, how quickly it arrives, and how much of the shared radio channel it consumes. This decision is the core of the project.

3.3 Importance of Sockets, UDP and Broadcast

The application uses UDP for three reasons. First, an alert is time-critical and short; a TCP connection set-up would add a round-trip of delay before any data could be sent, and TCP retransmissions would deliver stale information. Second, the sender does not know which vehicles are nearby, so it cannot open connections to them; a UDP datagram sent to the subnet broadcast address 10.1.255.255 is delivered to every vehicle in radio range. Third, UDP leaves reliability to the application, which is exactly where the relaying logic lives.

The socket operations used are the same ones used in C on Linux: the application creates a datagram socket, enables broadcast (the equivalent of the `SO_BROADCAST` option), binds the socket to port 4040 on all local addresses, registers a callback that runs when a datagram arrives, and sends with `SendTo()` to the broadcast address. In ns-3 these operations are provided by the `ns3::Socket` class, which follows the Berkeley API closely.

3.4 Tools and Environment

- **ns-3.45:** an open-source discrete-event network simulator written in C++. It models the full protocol stack (applications, sockets, UDP, IPv4, the 802.11 MAC and PHY, and radio propagation) and is widely used in networking research.
- **C++20:** the simulation program, the socket application and the custom packet header are written in C++ against the ns-3 API.
- **IEEE 802.11p model of the ns-3 wifi module:** 10 MHz OFDM channel, ad hoc MAC.
- **Python 3 with matplotlib:** scripts that run the 180-run parameter sweep in parallel, aggregate the results and draw the graphs.
- A browser-based visualizer (HTML and JavaScript) that replays the packet trace written by the simulator.
- **Hardware and OS:** an 8-core Apple Silicon laptop with 16 GB RAM running macOS. The same code builds unchanged on Ubuntu Linux or Ubuntu under WSL.

CHAPTER 4

LITERATURE REVIEW

Sixteen peer-reviewed papers from IEEE, ACM, Elsevier and Springer venues were studied. They are grouped below by theme, and Table 4.1 summarises each paper's approach and its limitation with respect to our problem.

4.1 The Broadcast Storm Problem

Ni, Tseng, Chen and Sheu [1] first identified the broadcast storm problem in mobile ad hoc networks. They showed analytically that when a node re-broadcasts a packet it has just received, the extra area it covers is at most 61 % of its radio range and only about 41 % on average, because its range overlaps heavily with the sender's. Blind flooding therefore produces many redundant transmissions, heavy contention for the channel and frequent collisions. They proposed probabilistic, counter-based, distance-based, location-based and cluster-based schemes to suppress redundant re-broadcasts. Their analysis assumes a two-dimensional MANET and does not consider the linear geometry or the direction of travel of a road.

Wisitpongphan, Tonguz and colleagues [2] studied the broadcast storm specifically in VANETs, measuring its effect on packet loss and delay on a highway, and proposed three suppression schemes for DSRC: weighted p-persistence, slotted 1-persistence and slotted p-persistence. In all three, receivers farther from the sender are given a higher probability or an earlier time slot to relay. These schemes reduced packet loss by up to about 70 %. They decide only on distance and do not check whether a vehicle further ahead has already relayed.

4.2 Distance-Based and Contention-Based Broadcast

Korkmaz et al. [3] proposed the Urban Multi-hop Broadcast protocol (UMB), which divides the sender's range into segments and selects one vehicle in the farthest non-empty segment as the forwarder using a request/clear-to-broadcast handshake and an acknowledgement. Fasolo, Zanella and Zorzi [4] proposed Smart Broadcast, in which receivers choose a back-off according to their position within the sender's range so that the farthest vehicles tend to relay first, and derived optimal parameters analytically. Tonguz, Wisitpongphan and Bai [5] proposed DV-CAST, which uses periodic hello messages to learn the local topology and switches between broadcast suppression in dense

traffic and store-carry-forward in sparse or disconnected traffic. Füßler et al. [6] proposed contention-based forwarding, in which neighbours start timers that are shorter the more progress they offer and the first to expire forwards; they showed that suppressing duplicate forwarders is the hardest part, because nodes that cannot hear each other produce duplicates.

These works establish that letting the farthest receiver relay first is effective. However, UMB and the active-selection variant of contention-based forwarding depend on MAC-level handshakes or acknowledgements that plain 802.11p broadcast does not provide, DV-CAST depends on beacons for neighbour knowledge, and Smart Broadcast assumes an idealised channel.

4.3 Probabilistic Dissemination

Haas, Halpern and Li [7] replaced flooding with gossip, in which each node forwards with a fixed probability p . They showed a bimodal behaviour: the message either dies out early or reaches almost all nodes, and values of p between 0.6 and 0.8 reach nearly all nodes with up to 35 % fewer messages than flooding in large networks. Zeng, Yu and Wang [8] made the forwarding probability depend on distance and added clustering to reduce redundancy, deriving the expected delay and number of copies. Fixed-probability gossip is simple but fragile when the network is thin, as on a highway where each car has only a few neighbours in the direction the alert must travel.

4.4 IEEE 802.11p and Channel Effects

Jiang and Delgrossi [9] described the IEEE 802.11p amendment, which allows vehicles to communicate outside the context of a basic service set, without association or authentication, and adapts the OFDM physical layer to a 10 MHz channel for fast-moving vehicles. Torrent-Moreno, Jiang and Hartenstein [10] measured broadcast reception rates in 802.11-based VANETs and showed that under Nakagami fading the probability of receiving a broadcast drops sharply compared with deterministic propagation models, so that repetition or multi-hop re-transmission is needed for reliable warnings. This finding directly motivates our use of a fading channel, background beacon load and periodic repetition of the alert.

4.5 Surveys and Recent Work

Hartenstein and Laberteaux [11] give a tutorial survey of VANETs, their applications and their open problems, including scalable broadcast of safety messages. Latif et al. [12] compare many scenario-driven multi-hop broadcast protocols and note that they are evaluated with different simulators and channel models, which makes them hard to compare. Shah et al. [13] proposed a time-barrier scheme in which effectively only the farthest vehicle re-broadcasts, reducing overhead compared with plain broadcast and cluster-based schemes. Benrhaïem, Hafid and Sahu [14] proposed REMD, which estimates link quality from beacons and chooses the number of repetitions per hop to meet a reliability target. Cao et al. [15] identified cases in which pure distance-based relay selection fails and repaired them with a mini black-burst mechanism at the MAC layer. Raviglione et al. [16] presented ms-van3t, an ns-3 based framework for realistic evaluation of V2X communication with 802.11p, LTE-V2X and NR-V2X.

Table 4.1: Summary of the literature reviewed

Ref.	Approach	Key result	Limitation for our problem
[1] Ni et al., 1999	Defines the broadcast storm; probabilistic, counter, distance, location and cluster suppression	A relay adds at most 61 % (avg. 41 %) new coverage	2-D MANET; ignores road geometry and direction
[2] Wisitpongphan et al., 2007	Weighted p-persistence, slotted 1- and p-persistence for DSRC	Up to ~70 % lower packet loss	Distance only; no cancellation after a farther relay
[3] Korkmaz et al., 2004	UMB: farthest-segment forwarder chosen by RTB/CTB handshake	Reliable multi-hop broadcast in cities	Needs MAC handshake and ACKs
[4] Fasolo et al., 2006	Smart Broadcast: position-based back-off	Fast propagation with optimised parameters	Idealised channel, one direction
[5] Tonguz et al., 2010	DV-CAST: suppression plus store-carry-forward	Works in dense and disconnected traffic	Relies on periodic hello beacons
[6] Füller et al., 2003	Contention-based forwarding with progress timers	Beaconless greedy forwarding	Unicast; duplicates from hidden nodes
[7] Haas et al., 2006	Gossip with fixed forwarding probability p	$p = 0.6-0.8$ saves up to 35 % messages	Bimodal; dies out in thin networks
[8] Zeng et al., 2018	Distance-dependent probability with clustering	Analytical delay and copy count	Needs clusters; no directional cancellation
[9] Jiang & Delgrossi, 2008	IEEE 802.11p (WAVE) overview	Communication without association	Standard overview, no evaluation
[10] Torrent-Moreno et al., 2004	Broadcast reception under fading and EDCA priority	Fading sharply lowers reception	Recommends repetition; no relay scheme
[11] Hartenstein & Laberteaux, 2008	Tutorial survey of VANETs	Applications and open problems	Survey
[12] Latif et al., 2018	Comparative study of multi-hop broadcast protocols	Scenario-based classification	Mixed simulators and channel models
[13] Shah et al., 2019	Time-barrier relaying by the farthest vehicle	Lower overhead than plain broadcast	No direction-aware cancellation
[14] Benrhaïem et al., 2020	REMD: per-hop repetitions from beacon	Meets a target per-hop reliability	Beacon estimation and cell structure needed

Ref.	Approach	Key result	Limitation for our problem
	link estimates		
[15] Cao et al., 2020	Distance-based relay selection with mini black-burst	>99.99 % delivery, faster dissemination	Needs MAC-level black-burst
[16] Raviglione et al., 2024	ms-van3t: ns-3 + SUMO V2X framework	Full ETSI C-ITS stack, 802.11p to NR-V2X	Platform, no relay comparison

CHAPTER 5

RESEARCH GAP

The literature review reveals several gaps:

- **Distance without direction.** Distance-based deferral schemes [2], [4], [8], [13], [15] let farther receivers relay first, but they do not cancel a pending relay when a vehicle further along the direction of propagation has already relayed. Where cancellation exists [3], [6], it relies on MAC handshakes, acknowledgements or black-bursts.
- **Dependence on MAC changes or control traffic.** Many schemes need request/clear-to-broadcast handshakes, black-bursts, clustering or beacon-based link estimation [3], [8], [14], [15]. Very few work purely at the application layer over standard UDP broadcast on an unmodified 802.11p stack, which is what a socket programmer can actually deploy.
- **Idealised channels in evaluations.** Fading sharply reduces broadcast reception [10], yet relay schemes are seldom compared head-to-head under fading together with periodic beacon load; the literature mixes simulators and channel models [12].
- **Repetition and suppression studied separately.** Repetition is recommended [10] and tuned per hop [14], but its interaction with relay suppression is not measured.
- **Weak or inconsistent baselines.** Fixed-probability gossip [7] behaves bimodally, and distance-weighted variants [8] need clustering, so a plain, reproducible three-way comparison of flooding, gossip and distance-based contention is rare.

Selected research gap. Among these, this work addresses one gap: there is no simple, application-layer relaying scheme for emergency alerts that combines distance-based contention with direction-aware suppression, runs over standard UDP broadcast on IEEE 802.11p without any MAC changes or extra control messages, and has been evaluated against flooding and gossip under fading, background load and periodic alert repetition.

CHAPTER 6

PROBLEM STATEMENT

To design, implement and evaluate, using UDP socket programming in the ns-3 network simulator, an emergency alert broadcasting application for vehicular ad hoc networks in which a crashed vehicle periodically broadcasts an accident alert and every other vehicle relays it with a distance-based, direction-aware Smart Relay scheme implemented entirely at the application layer, so that the alert reaches at least as many vehicles as blind flooding with fewer transmissions and collisions, while keeping the end-to-end delay well below the 100 ms required for safety warnings, over an IEEE 802.11p channel with fading and background beacon traffic.

CHAPTER 7

METHODOLOGY AND IMPLEMENTATION

7.1 Proposed System Architecture

Figure 7.1 shows the architecture of the system. Every vehicle is an ns-3 node carrying the full protocol stack. At the top, two applications run on every vehicle. VehicleApp is the emergency alert application: on the crashed vehicle it generates an alert every second, and on every vehicle it receives alerts, keeps a duplicate cache keyed by (origin, sequence number), warns the driver on the first copy, and runs the relay engine that decides whether to re-broadcast. BeaconApp sends five 200-byte status beacons per second, which load the channel the way periodic cooperative-awareness messages do in real deployments.

Below the applications, each vehicle uses a UDP socket, IPv4 with subnet broadcast, the IEEE 802.11p ad hoc MAC and the OFDM physical layer. All vehicles share one wireless channel whose model includes log-distance path loss and Nakagami fading. Trace sinks attached to the mobility model and the physical layer record every movement, every reception and every lost frame; these records produce the metrics, the graphs and the animated replay.

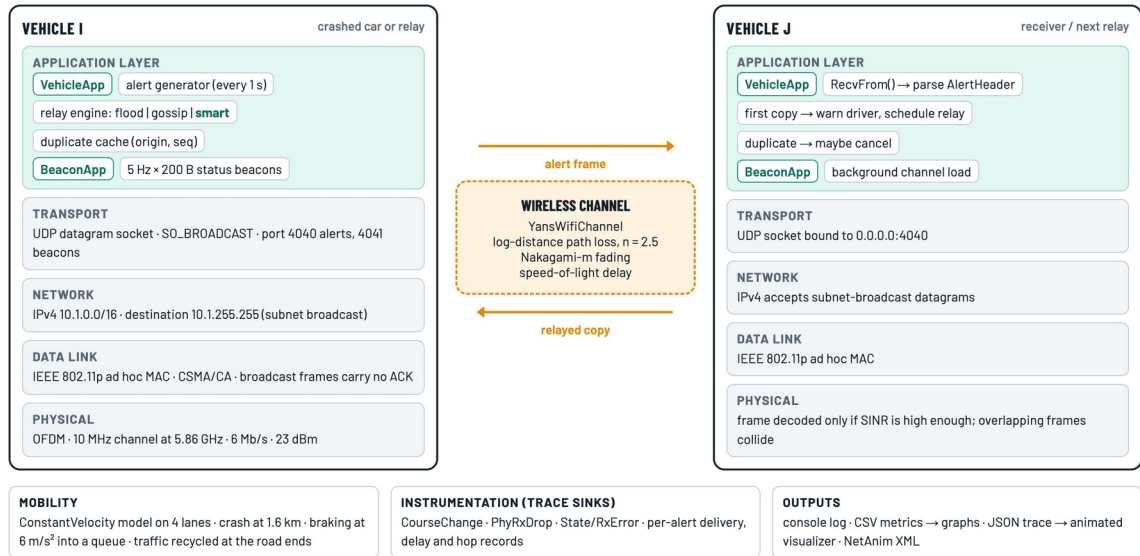


Figure 7.1: System architecture of the Smart Emergency Alert Broadcasting application

7.2 Network Communication Model

The scenario is shown in Figure 7.2. A straight 2 km highway has four lanes, two in each direction. Since traffic keeps left in India, eastbound vehicles occupy the upper half of the

road. Eastbound lanes carry traffic at 30 m/s and 25 m/s, westbound lanes at 31 m/s and 26 m/s. Vehicles are placed randomly with a minimum gap of 15 m in each lane, and vehicles that leave one end of the section re-enter at the other, so that the traffic density stays constant.

At $t = 3$ s, Vehicle 0 crashes at $x = 1,600$ m and stops. Both eastbound lanes are blocked behind it. Eastbound vehicles that are still behind the crash brake at 6 m/s^2 and stop in a queue with 10 m spacing. Vehicle 0 broadcasts ACCIDENT ALERT number 1 at $t = 3$ s and repeats it with a new sequence number every second until the end of the 20 s simulation, giving 17 alerts per run. Each alert is a separate dissemination: every vehicle that receives it decides independently whether to relay it.

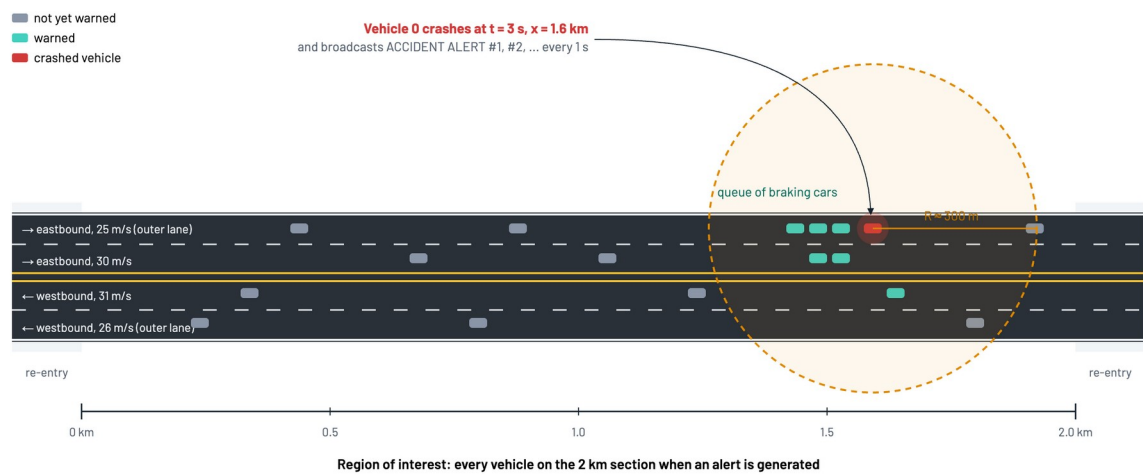


Figure 7.2: Highway scenario: crash location, radio range and region of interest

Communication model. All vehicles belong to one IPv4 subnet, 10.1.0.0/16. The crashed vehicle sends each alert as one UDP datagram to 10.1.255.255:4040. The IP layer maps the broadcast address to the 802.11 broadcast MAC address, so the frame is received by every vehicle within radio range. Broadcast frames are not acknowledged and are never retransmitted by the MAC, so a frame lost to a collision or to fading is simply lost; only a relay by another vehicle, or the next periodic alert, can make up for it. Table 7.1 lists the simulation parameters.

Table 7.1: Simulation parameters

Parameter	Value
Simulator	ns-3.45 (C++20, optimized build)
Road	2,000 m straight highway, 4 lanes (2 per direction), 3.5 m lanes
Vehicles	20, 40, 60, 80, 100, 120 (density sweep); 60 in examples
Lane speeds	Eastbound 30 and 25 m/s; westbound 31 and 26 m/s
Mobility	Constant velocity; braking at 6 m/s^2 into a queue;

Parameter	Value
	re-entry at the road ends
Crash	Vehicle 0 at $x = 1,600$ m, $t = 3$ s
Alerts	One every 1 s from $t = 3$ s; 17 scored per run; simulation time 20 s
Alert datagram	111 B UDP payload (40 B AlertHeader + 71 B text); 175 B on air
Transport / network	UDP to 10.1.255.255:4040 (alerts), :4041 (beacons); IPv4 10.1.0.0/16
MAC	IEEE 802.11p ad hoc; broadcast without ACK or retransmission; CWmin = 15
PHY	OFDM, 10 MHz channel at 5.86 GHz, 6 Mb/s, 23 dBm transmit power
Propagation	Log-distance ($n = 2.5$, $L_0 = 47.86$ dB) + Nakagami-m ($m = 1.5 / 0.75 / 0.75$)
Background load	5 beacons/s of 200 B per vehicle
Smart Relay	$R = 300$ m, $N = 5$ slots, $\tau = 2$ ms, jitter $U(0, 0.5$ ms), max 20 hops
Gossip	Forwarding probability $p = 0.6$
Repetitions	10 seeds per point (RngRun 1–10), 180 runs in total

7.3 Mathematical Model

Radio propagation. The received power at distance d from a transmitter is modelled with the log-distance path loss model, followed by Nakagami-m fading:

$$Prx(d) = Ptx - L_0 - 10 \cdot n \cdot \log_{10}(d / d_0) \quad (7.1)$$

where $Ptx = 23$ dBm, $L_0 = 47.86$ dB is the free-space loss at $d_0 = 1$ m at 5.9 GHz, and $n = 2.5$. With fading, the instantaneous received power is drawn from a Gamma distribution with the mean given by (7.1) and shape $m = 1.5$ below 80 m and $m = 0.75$ beyond, so a link that is good on average can still fail for an individual frame. A frame is decoded only if its signal-to-interference-plus-noise ratio is high enough for the 6 Mb/s mode; two overlapping frames at a receiver usually destroy each other.

Frame air time. One alert frame is 175 bytes on air (24 B MAC header, 8 B LLC, 20 B IPv4, 8 B UDP, 40 B AlertHeader, 71 B text, 4 B FCS). At 6 Mb/s on a 10 MHz channel, each OFDM symbol lasts 8 μ s and carries 48 data bits, so the frame needs

$$T_{frame} = 40 \mu s + \lceil (16 + 8 \cdot 175 + 6) / 48 \rceil \cdot 8 \mu s = 40 + 30 \cdot 8 = 280 \mu s \quad (7.2)$$

where 40 μ s is the preamble and SIGNAL field. Adding the DIFS of 58 μ s gives an expected one-hop delay of about 0.34 ms when the channel is idle.

Contention in flooding. If k neighbours receive the same frame and all relay it immediately, each draws a random back-off of 0 to $CW = 15$ slots. The probability that a given relay picks a slot that no other relay has chosen is

$$P_{\text{success}} = (1 - 1 / (CW + 1))^{k-1} \quad (7.3)$$

With $k = 20$ neighbours this is $(15/16)^{19} \approx 0.29$, so about 70 % of the relays collide with another one. Hidden vehicles that cannot sense each other make it worse. This is the broadcast storm.

Smart Relay contention timer. A vehicle that receives the first copy of an alert from a sender at distance d waits

$$\text{slot} = \lfloor N \cdot (1 - \min(d, R) / R) \rfloor, \quad T_{\text{wait}} = \text{slot} \cdot \tau + U(0, \delta) \quad (7.4)$$

with nominal range $R = 300$ m, $N = 5$ slots, slot length $\tau = 2$ ms and jitter $\delta = 0.5$ ms. Vehicles near the edge of the sender's range (d close to R) get slot 0 and relay almost at once; vehicles close to the sender wait up to 8.5 ms. On a road, the new stretch of road covered by a relay at distance d is proportional to d , so the vehicle that extends coverage the most relays first.

Directional suppression. A pending vehicle at position x_{me} , whose relay direction is $\text{dir} = \text{sign}(x_{me} - x_{\text{sender}})$, cancels its relay if it hears a duplicate from a vehicle at position x_s with

$$(x_s - x_{me}) \cdot \text{dir} \geq 0 \quad (7.5)$$

that is, from a vehicle at or beyond it in the direction the alert is travelling. Such a vehicle has already covered everything this vehicle could cover. Duplicates arriving from behind are ignored, so a vehicle is never silenced by the transmission that reached it in the first place.

7.4 Proposed Algorithm

The Smart Relay algorithm runs inside VehicleApp's receive callback. Algorithm 1 gives it in pseudocode, Figure 7.3 shows it as a flowchart, and Figure 7.4 shows one round of dissemination as a sequence diagram.

```

Algorithm 1: Smart Relay (runs on every vehicle)
On receiving datagram p on UDP port 4040:
  h <- p.RemoveHeader(AlertHeader)
  if h.originId = myId or h.senderId = myId: discard
  key <- (h.originId, h.seq)
  if key not in cache:                                     // first copy
    deliver warning to driver; record delay and hop count
    d <- distance(myPos, (h.senderX, h.senderY))
    dir <- +1 if myPos.x >= h.senderX else -1
    slot <- floor(N * (1 - min(d, R) / R))
    cache[key] <- PENDING(dir, h)
    schedule Relay(key) after slot * tau + Uniform(0, delta)
  else if cache[key] is PENDING:                          // duplicate
    if (h.senderX - myPos.x) * dir >= 0:
      cancel scheduled Relay(key); cache[key] <- SUPPRESSED
Relay(key):
  h' <- copy of stored header; h'.senderId <- myId
  h'.senderX, h'.senderY <- myPos; h'.hops <- h.hops + 1
  SendTo(h' + text, 10.1.255.255:4040); cache[key] <- DONE

```

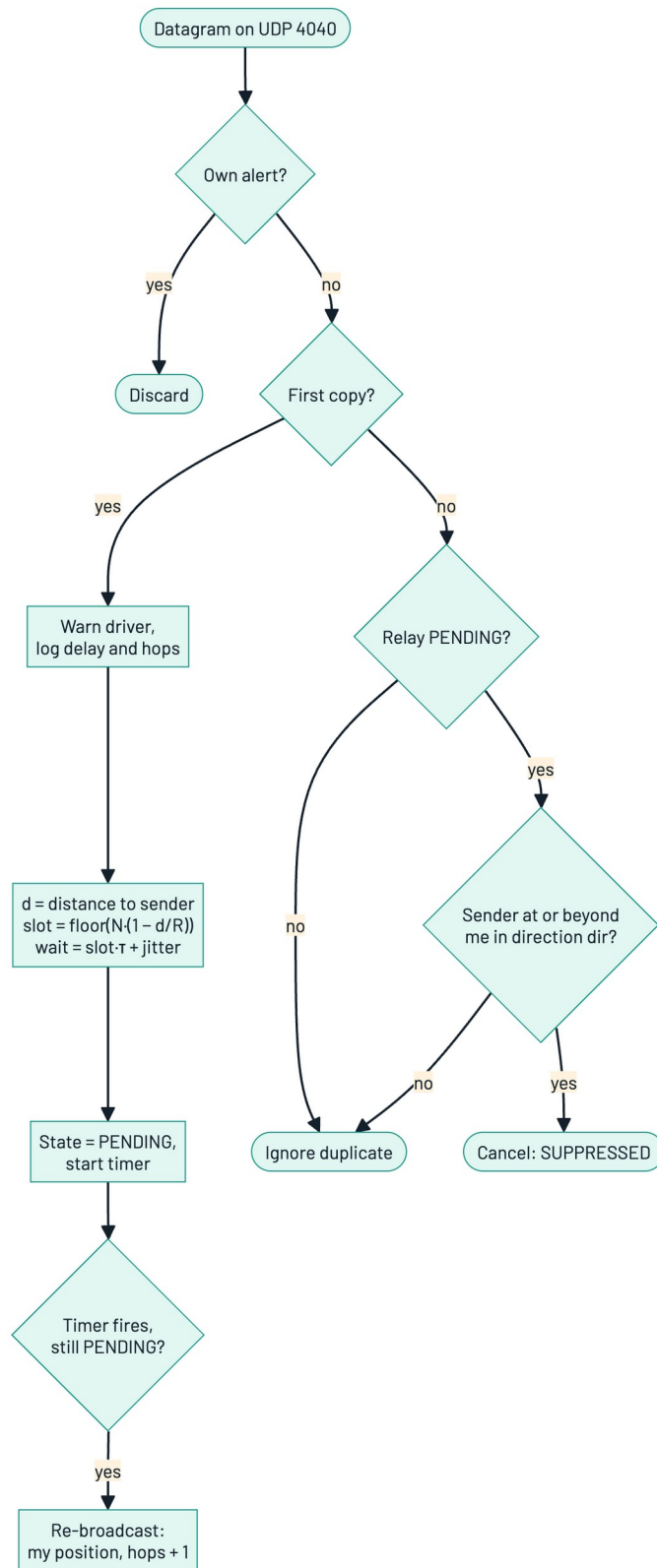


Figure 7.3: Flowchart of the Smart Relay receive and relay logic

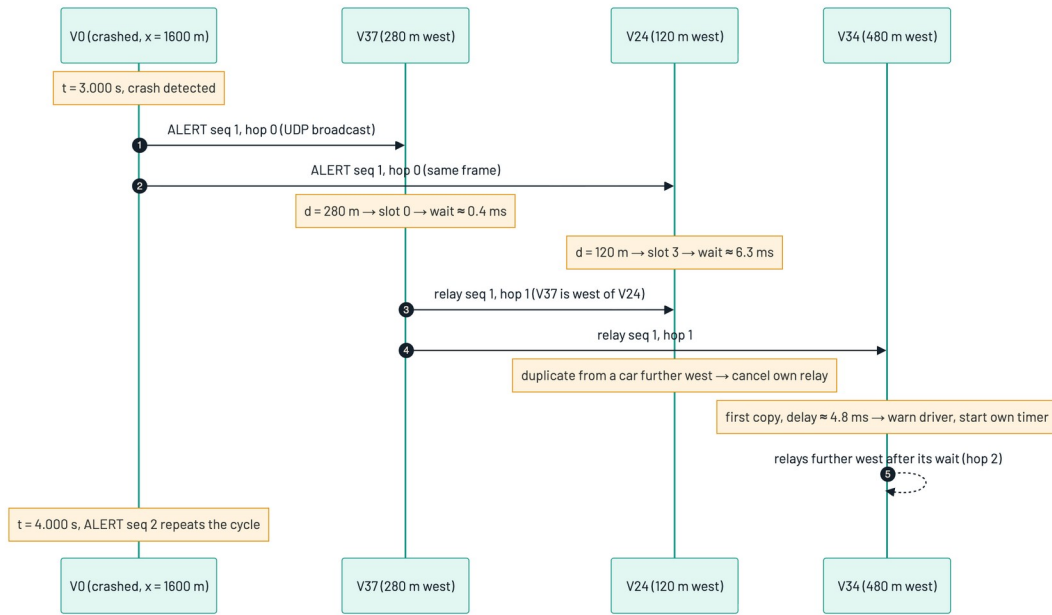


Figure 7.4: Sequence diagram of one alert dissemination round (illustrative distances)

Baseline schemes. The two baselines run in the same application with the same socket code; only the relay decision differs. In blind flooding, every vehicle relays immediately on the first copy. In gossip, every vehicle relays immediately on the first copy with probability $p = 0.6$ and otherwise drops it. Neither baseline looks at duplicates.

7.5 Socket Programming Implementation

The complete program, `vanet-smart-alert.cc`, is about 1,100 lines of C++. The socket-level parts are shown here.

Creating, configuring and binding the socket. Each `VehicleApp` creates a UDP socket when the application starts, enables broadcast and binds to port 4040 on all local addresses, then registers `HandleRead` as the receive callback:

```

m_socket = Socket::CreateSocket(GetNode(), UdpSocketFactory::GetTypeId());
m_socket->SetAllowBroadcast(true); // SO_BROADCAST
if (m_socket->Bind(InetSocketAddress(Ipv4Address::GetAny(), ALERT_PORT)) == -1)
{
    NS_FATAL_ERROR("Vehicle " << m_id << ": could not bind alert socket");
}
m_socket->SetRecvCallback(MakeCallback(&VehicleApp::HandleRead, this));

```

Sending an alert. An alert is built from the text payload, the 40-byte `AlertHeader` is prepended, and the datagram is sent to the subnet broadcast address:

```

Ptr<Packet> p = Create<Packet>(reinterpret_cast<const uint8_t*>(ALERT_TEXT),
                               static_cast<uint32_t>(std::strlen(ALERT_TEXT)));
p->AddHeader(h); // AlertHeader, 40 bytes
m_socket->SendTo(p, 0,
                InetSocketAddress(Ipv4Address("10.1.255.255"), ALERT_PORT));

```

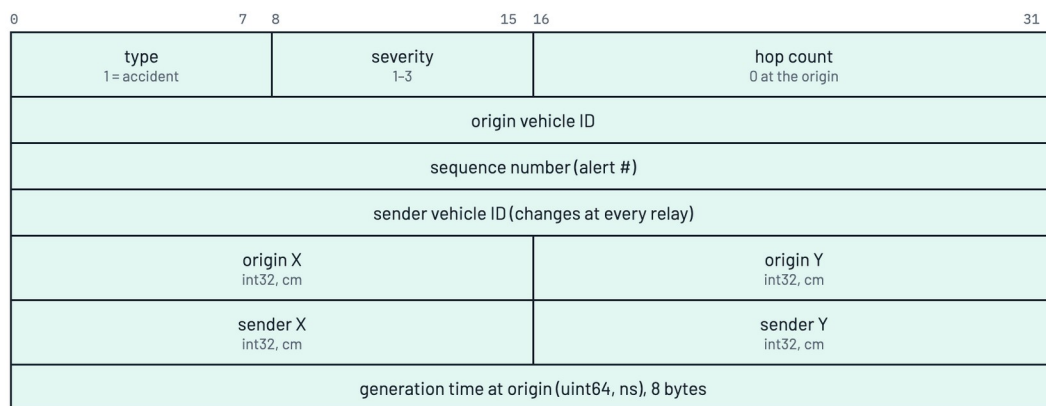
Receiving. HandleRead drains the socket with RecvFrom(), removes and parses the header, rejects malformed datagrams and the vehicle's own echoes, and passes the alert to the relay logic:

```
void HandleRead(Ptr<Socket> socket)
{
    Address from;
    while (Ptr<Packet> packet = socket->RecvFrom(from))
    {
        AlertHeader h;
        if (packet->GetSize() < h.GetSerializedSize())
            continue; // malformed datagram
        packet->RemoveHeader(h);
        if (h.originId == m_id || h.senderId == m_id)
            continue; // our own echo
        OnAlert(h, packet);
    }
}
```

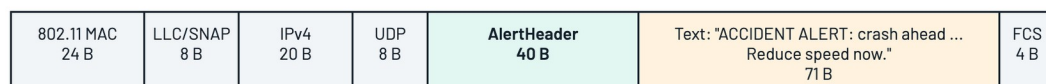
Application-layer header. AlertHeader is a subclass of ns3::Header and is serialised in network byte order, exactly as a C program would use htons() and htonl(). Figure 7.5 shows its layout and the encapsulation of a complete alert frame.

```
void Serialize(Buffer::Iterator i) const override
{
    i.WriteU8(type);           i.WriteU8(severity);           i.WriteHtonU16(hops);
    i.WriteHtonU32(originId); i.WriteHtonU32(seq); i.WriteHtonU32(senderId);
    i.WriteHtonU32(ToCm(originX)); i.WriteHtonU32(ToCm(originY));
    i.WriteHtonU32(ToCm(senderX)); i.WriteHtonU32(ToCm(senderY));
    i.WriteHtonU64(static_cast<uint64_t>(genTimeNs));
}
```

ALERTHEADER (40 BYTES, NETWORK BYTE ORDER)



Positions are 32-bit words each; they are drawn two per row to keep the figure compact. (origin, sequence) identifies an alert for duplicate detection; sender X/Y drives Smart Relay's distance and direction test.



Encapsulation of one alert frame on air (175 bytes).

Figure 7.5: AlertHeader format and encapsulation of one alert frame

Error handling. The program validates its command-line parameters (scheme name, number of vehicles, crash time and position, gossip probability, slot count, alert interval

and range) and exits with a clear message if any is invalid; it stops with an error if a socket cannot be bound, and it ignores datagrams shorter than the header.

CHAPTER 8

PERFORMANCE EVALUATION

8.1 Experimental Setup

The program was built with the optimized profile of ns-3.45 and run on an 8-core Apple Silicon laptop with 16 GB RAM. A Python script runs the full parameter sweep in parallel: six traffic densities (20, 40, 60, 80, 100 and 120 vehicles on the 2 km section), three schemes and ten random seeds per point, a total of 180 independent runs, which completed in 1 minute 41 seconds. Figure 8.1 shows the workflow.



Figure 8.1: Experimental workflow from the ns-3 program to graphs and the animated replay

Every random stream in the simulation (vehicle placement, MAC back-off, fading, beacon timing and relay jitter) is assigned a fixed stream number. As a result, for a given seed the three schemes see exactly the same vehicles, the same movements and the same channel conditions; the only difference between them is the relay rule. Each plotted point is the mean of ten seeds with a 95 % confidence interval.

Region of interest. An alert is scored over the vehicles that are on the 2 km section at the moment it is generated. Vehicles in the 150 m re-entry zones beyond each end are sparse edge cases and are excluded for all schemes alike. Each run scores the 17 alerts generated between $t = 3$ s and $t = 19$ s.

8.2 Performance Metrics

For a set of alerts A , let T_a be the vehicles in the region of interest when alert a is generated, R_a the vehicles that received it, $t_{\text{gen}}(a)$ its generation time and $\text{trx}(a, v)$ the time vehicle v first received it. Table 8.1 defines the metrics.

Table 8.1: Performance metrics

Metric	Definition	Unit
Packet delivery ratio (PDR)	$(1/ A) \sum_a R_a \cap T_a / T_a $	%
Complete coverage	Share of alerts with $R_a \supseteq T_a$	%
End-to-end delay	Mean of $\text{trx}(a, v) - t_{\text{gen}}(a)$ over all first receptions in T_a	ms
95th-percentile delay	95th percentile of the same delays	ms
Jitter	Mean of $ Dv(ak) - Dv(ak-1) $	ms

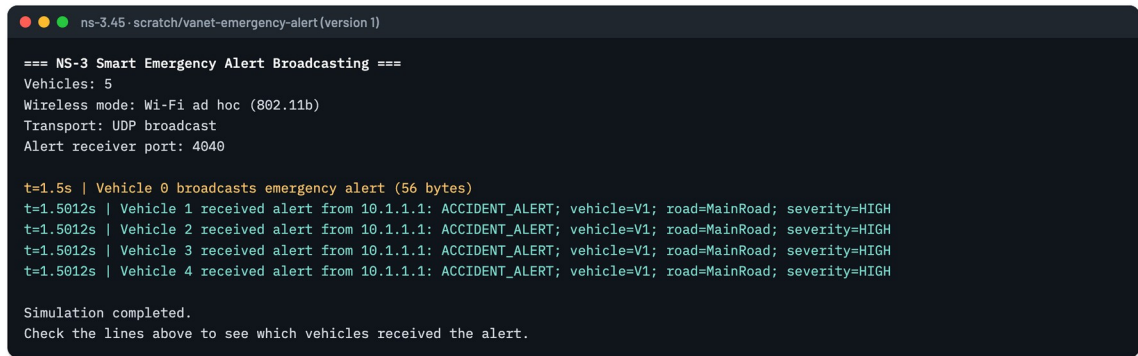
Metric	Definition	Unit
	over vehicles v and consecutive alerts	
Goodput (throughput)	$\sum_a Ra \cap Ta \cdot S \cdot 8 / (A \cdot \Delta)$, $S = 111 \text{ B}$, $\Delta = 1 \text{ s}$	kb/s
Transmissions per alert	(origin + relay transmissions) / $ A $	count
Re-broadcast ratio	Relay transmissions / first receptions	%
Lost frames per alert	Alert frames dropped or undecodable at receivers / $ A $	count
Hop count	Hops travelled by the first copy to reach each vehicle	hops

CHAPTER 9

OUTPUT AND RESULTS

9.1 Program Output

Figure 9.1 shows the output of the first version of the program, which places five parked vehicles 10 m apart on an 802.11b ad hoc network and sends one broadcast. All four receivers receive the alert at $t = 1.5012$ s. The 1.2 ms delay matches theory: a 120-byte frame at 1 Mb/s takes 0.96 ms, plus a 192 μ s preamble and a 50 μ s DIFS, giving 1.2 ms.



```

ns-3.45 - scratch/vanet-emergency-alert (version 1)

=== NS-3 Smart Emergency Alert Broadcasting ===
Vehicles: 5
Wireless mode: Wi-Fi ad hoc (802.11b)
Transport: UDP broadcast
Alert receiver port: 4040

t=1.5s | Vehicle 0 broadcasts emergency alert (56 bytes)
t=1.5012s | Vehicle 1 received alert from 10.1.1.1: ACCIDENT_ALERT; vehicle=V1; road=MainRoad; severity=HIGH
t=1.5012s | Vehicle 2 received alert from 10.1.1.1: ACCIDENT_ALERT; vehicle=V1; road=MainRoad; severity=HIGH
t=1.5012s | Vehicle 3 received alert from 10.1.1.1: ACCIDENT_ALERT; vehicle=V1; road=MainRoad; severity=HIGH
t=1.5012s | Vehicle 4 received alert from 10.1.1.1: ACCIDENT_ALERT; vehicle=V1; road=MainRoad; severity=HIGH

Simulation completed.
Check the lines above to see which vehicles received the alert.

```

Figure 9.1: Output of version 1: single-hop UDP broadcast to four parked vehicles

Figure 9.2 shows the output of the final program with Smart Relay and 60 vehicles. The crash is announced at $t = 3.000$ s, the alert is broadcast to 10.1.255.255:4040, and the first nine vehicles receive it after 0.34 ms, exactly the one-hop delay predicted by Equation (7.2). Vehicle 37, the first receiver whose contention timer expires, relays it, and Vehicles 34, 45 and 46 receive it at hop 2 after 4.77 ms. The summary at the end of the run reports a delivery ratio of 98.71 % for this seed.

```

ns-3.45 - vanet-smart-alert -scheme=smart -vehicles=60 -verbose=1

=== NS-3 Smart Emergency Alert Broadcasting (VANET) ===
Scheme      : smart
Vehicles    : 60 on a 2000 m, 4-lane highway
Radio       : IEEE 802.11p ad hoc, 10 MHz, 6 Mb/s, 23 dBm
Transport   : UDP broadcast 10.1.255.255:4040
Crash       : t=3 s at x=1600 m, alert every 1 s
Run (seed)  : 1

t=3.000s | *** Vehicle 0 CRASHED at x=1600.0 m, eastbound carriageway blocked ***
t=3.000s | Vehicle 0 broadcasts ACCIDENT ALERT #1 to 10.1.255.255:4040
t=3.0003s | Vehicle 24 got alert #1 from Vehicle 0 (hop 1, delay 0.34 ms): ACCIDENT ALERT: crash ahead on eastbound carriageway. Reduce speed now.
t=3.0003s | Vehicle 9 got alert #1 from Vehicle 0 (hop 1, delay 0.34 ms): ACCIDENT ALERT: crash ahead on eastbound carriageway. Reduce speed now.
t=3.0003s | Vehicle 47 got alert #1 from Vehicle 0 (hop 1, delay 0.34 ms): ACCIDENT ALERT: crash ahead on eastbound carriageway. Reduce speed now.
t=3.0003s | Vehicle 50 got alert #1 from Vehicle 0 (hop 1, delay 0.34 ms): ACCIDENT ALERT: crash ahead on eastbound carriageway. Reduce speed now.
t=3.0003s | Vehicle 4 got alert #1 from Vehicle 0 (hop 1, delay 0.34 ms): ACCIDENT ALERT: crash ahead on eastbound carriageway. Reduce speed now.
t=3.0003s | Vehicle 33 got alert #1 from Vehicle 0 (hop 1, delay 0.34 ms): ACCIDENT ALERT: crash ahead on eastbound carriageway. Reduce speed now.
t=3.0003s | Vehicle 15 got alert #1 from Vehicle 0 (hop 1, delay 0.34 ms): ACCIDENT ALERT: crash ahead on eastbound carriageway. Reduce speed now.
t=3.0003s | Vehicle 5 got alert #1 from Vehicle 0 (hop 1, delay 0.34 ms): ACCIDENT ALERT: crash ahead on eastbound carriageway. Reduce speed now.
t=3.0003s | Vehicle 37 got alert #1 from Vehicle 0 (hop 1, delay 0.34 ms): ACCIDENT ALERT: crash ahead on eastbound carriageway. Reduce speed now.
t=3.0048s | Vehicle 34 got alert #1 from Vehicle 37 (hop 2, delay 4.77 ms): ACCIDENT ALERT: crash ahead on eastbound carriageway. Reduce speed now.
t=3.0048s | Vehicle 45 got alert #1 from Vehicle 37 (hop 2, delay 4.77 ms): ACCIDENT ALERT: crash ahead on eastbound carriageway. Reduce speed now.
t=3.0048s | Vehicle 46 got alert #1 from Vehicle 37 (hop 2, delay 4.77 ms): ACCIDENT ALERT: crash ahead on eastbound carriageway. Reduce speed now.
... (922 reception lines in total, trimmed)

--- Results (smart, 60 vehicles, run 1) ---
Alerts scored      : 17
Packet delivery ratio : 98.71 %
Alerts reaching every car : 64.71 %
Mean / 95th pct delay : 9.16 / 29.08 ms
Jitter            : 6.16 ms
Mean / max hops    : 3.69 / 11
Transmissions per alert : 26.06
Re-broadcast ratio  : 46.20 %
Lost alert frames / alert : 37.47
Alert goodput      : 43.56 kb/s
Queued, not eyewitnesses : 5 (0 braked without a warning)
Average warning lead time : 11.85 s

```

Figure 9.2: Output of the final program with Smart Relay (60 vehicles, seed 1, trimmed)

Figure 9.3 shows the summary of blind flooding on the same traffic. Flooding sends 51 transmissions per alert, nearly twice Smart Relay's 26, and loses 1,039 alert frames per alert to collisions against Smart Relay's 37, yet still reaches fewer vehicles (91.85 % against 98.71 %).

```

vanet-smart-alert -scheme=flood -vehicles=60 (same traffic, seed 1)

--- Results (flood, 60 vehicles, run 1) ---
Alerts scored      : 17
Packet delivery ratio : 91.85 %
Alerts reaching every car : 35.29 %
Mean / 95th pct delay : 1.63 / 3.40 ms
Jitter            : 0.40 ms
Mean / max hops    : 4.42 / 11
Transmissions per alert : 51.00
Re-broadcast ratio  : 100.00 %
Lost alert frames / alert : 1038.76
Alert goodput      : 40.48 kb/s
Queued, not eyewitnesses : 5 (0 braked without a warning)
Average warning lead time : 11.85 s

```

Figure 9.3: Summary output of blind flooding on the same traffic (seed 1)

9.2 Visual Replay of the Simulation

The simulator writes a trace of every movement, transmission, reception, scheduled relay, cancelled relay and lost frame. The browser-based visualizer replays the three schemes side by side on the same traffic, and slows down automatically during each alert wave so that the hop-by-hop spread is visible. Figures 9.4 and 9.5 are frames from this replay.

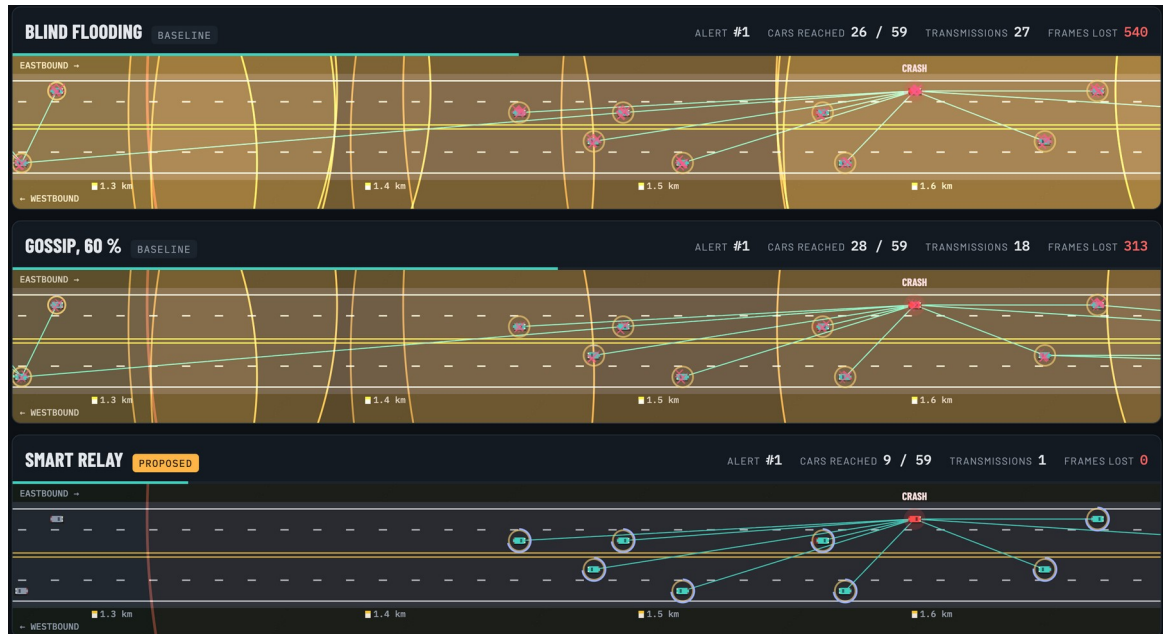


Figure 9.4: Crash site 1.5 ms after the crash: flooding has made 27 transmissions and lost 540 frames; Smart Relay has made one transmission while nearby cars wait their turn



Figure 9.5: The whole 2 km road 30 ms after the crash. Lines show which car each vehicle received the alert from

In Figure 9.4, every car around the crash in the flooding strip has received the alert and is transmitting at the same moment; the overlapping radio rings and the red marks are the broadcast storm. In the Smart Relay strip only the crash vehicle has transmitted, and the circular countdowns show the cars waiting, with the farthest about to relay first. In Figure 9.5, gossip has reached only 28 of 59 vehicles: the alert died out in the middle of the road, the bimodal failure predicted in [7]. Smart Relay has reached 55 vehicles with 21 transmissions and 22 lost frames, against flooding's 57 transmissions and 1,132 lost frames.

9.3 Comparative Analysis

Table 9.1 gives the mean results at four densities, and Figures 9.6 to 9.13 plot each metric across all six densities.

Table 9.1: Mean results over 10 seeds (best value in bold)

Vehicles	Metric	Flooding	Gossip	Smart Relay
40	PDR (%)	53.3	44.8	78.4
	Mean delay (ms)	1.25	0.91	10.79
	Jitter (ms)	0.25	0.19	4.73
	Tx per alert	19.3	10.2	16.0
	Lost frames / alert	239.7	103.2	13.8
	Goodput (kb/s)	15.7	13.2	23.2
60	PDR (%)	71.0	56.4	96.0
	Mean delay (ms)	1.59	1.27	9.97
	Jitter (ms)	0.39	0.30	5.24
	Tx per alert	39.4	18.6	25.0
	Lost frames / alert	804.0	318.4	45.2
	Goodput (kb/s)	31.3	24.8	42.4
80	PDR (%)	84.6	64.5	98.8
	Mean delay (ms)	1.78	1.48	8.40
	Jitter (ms)	0.46	0.34	4.57
	Tx per alert	62.8	28.4	29.3
	Lost frames / alert	1,841.0	736.1	97.2
	Goodput (kb/s)	50.1	38.1	58.4
120	PDR (%)	93.9	83.2	99.3
	Mean delay (ms)	1.84	1.79	5.81
	Jitter (ms)	0.55	0.53	3.04
	Tx per alert	106.6	56.0	35.0
	Lost frames / alert	5,084.2	2,376.9	254.2
	Goodput (kb/s)	84.4	74.8	89.2

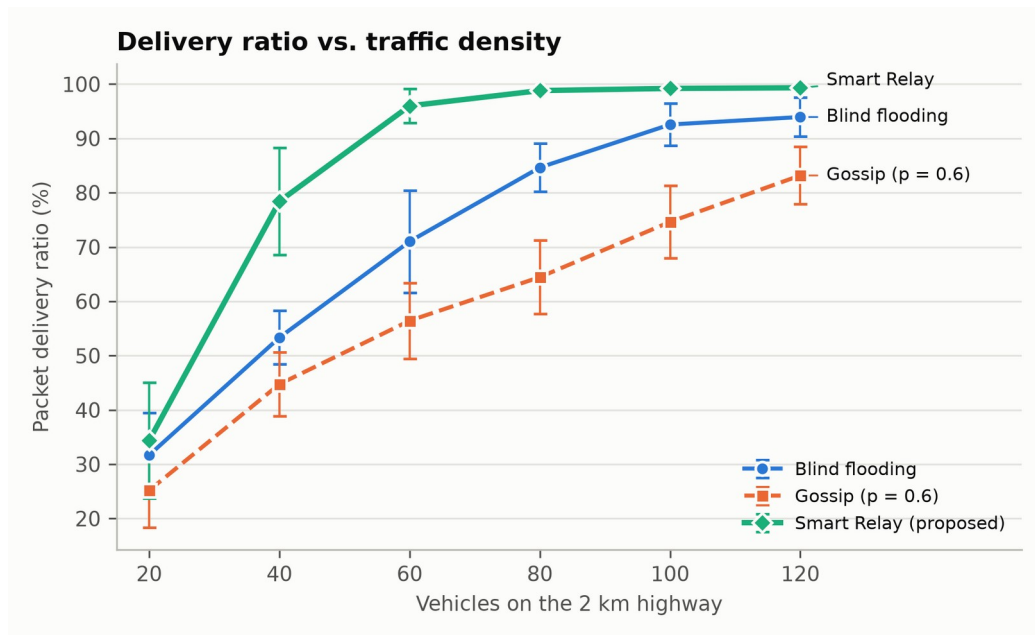


Figure 9.6: Packet delivery ratio against traffic density

Packet delivery ratio. Smart Relay delivers the most alerts at every density (Figure 9.6). The gap is largest at medium density: at 60 vehicles Smart Relay reaches 96.0 % against 71.0 % for flooding and 56.4 % for gossip. At 120 vehicles it reaches 99.3 % against 93.9 %. At 20 vehicles all schemes are low (25 % to 34 %) because with an average of one car every 400 m per lane the road is frequently disconnected; no relay rule can cross a gap wider than the radio range.

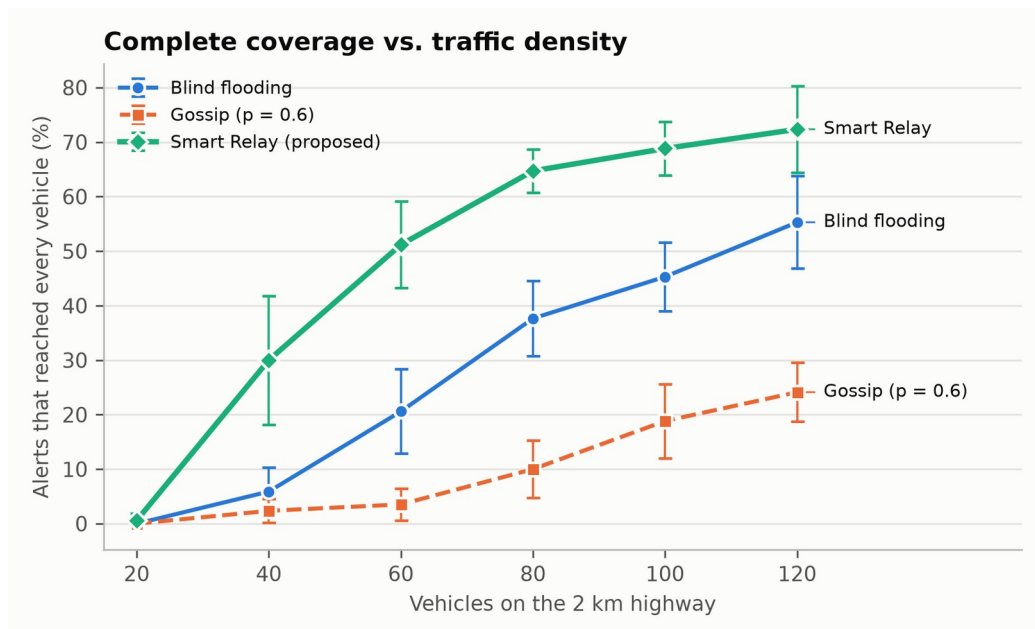


Figure 9.7: Percentage of alerts that reached every vehicle on the section

Complete coverage. The share of alerts that reached every single vehicle (Figure 9.7) rises to 72.4 % with Smart Relay at 120 vehicles, compared with 55.3 % for flooding and 24.1 % for gossip.

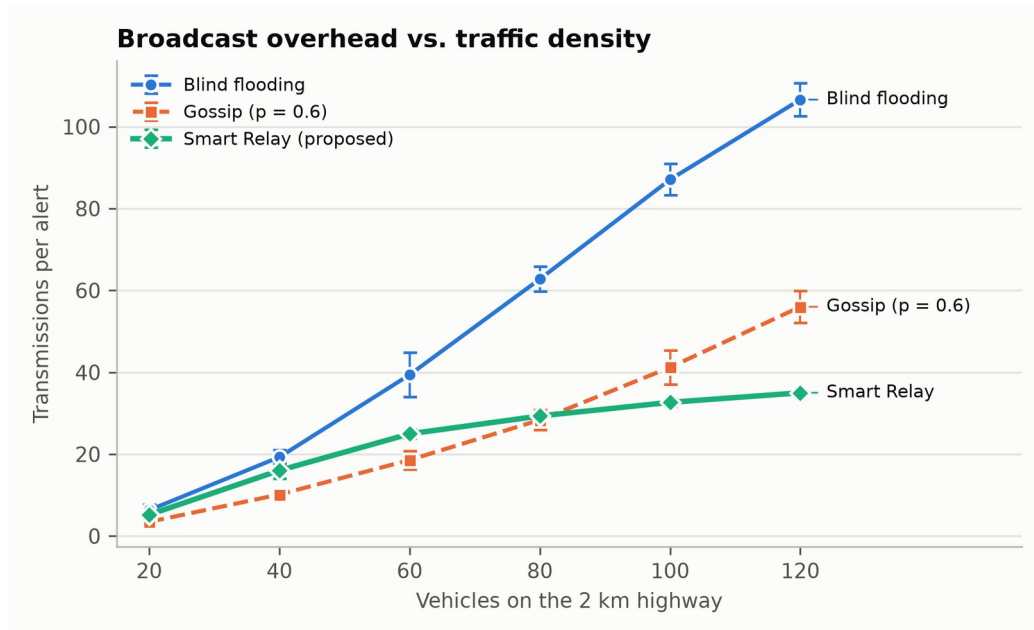


Figure 9.8: Transmissions needed per alert (broadcast overhead)

Overhead. Flooding's transmissions grow linearly with the number of vehicles, because every receiver transmits (Figure 9.8). Smart Relay's grow only slowly, from 16.0 at 40 vehicles to 35.0 at 120, because extra vehicles mostly cancel their relays. At 120 vehicles Smart Relay sends 67 % fewer packets than flooding. The share of receivers that re-broadcast falls from 74.7 % to 30.1 % as density increases (Figure 9.9).

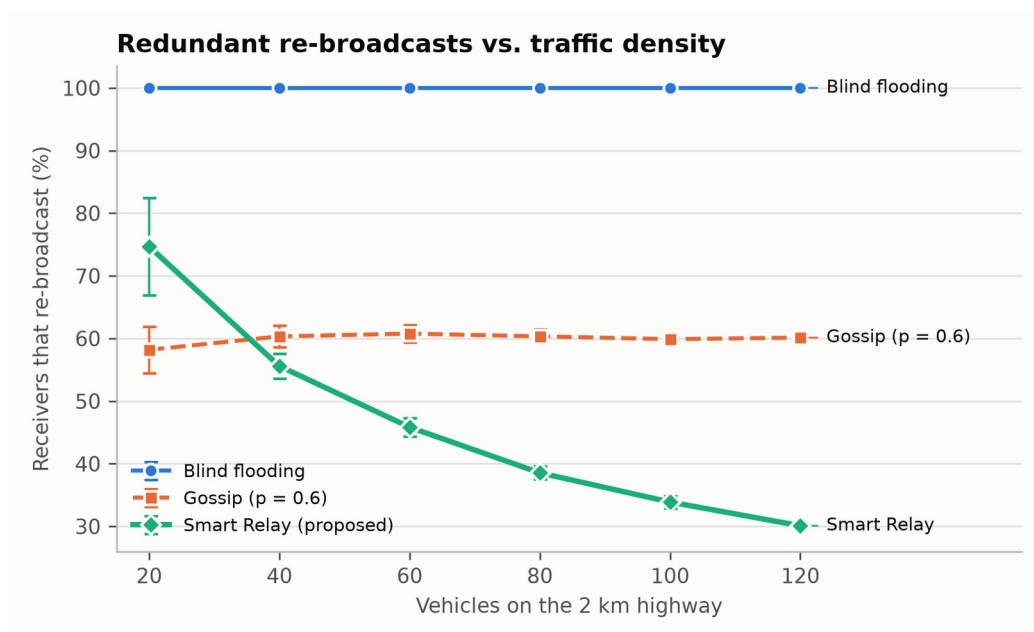


Figure 9.9: Share of receivers that re-broadcast the alert

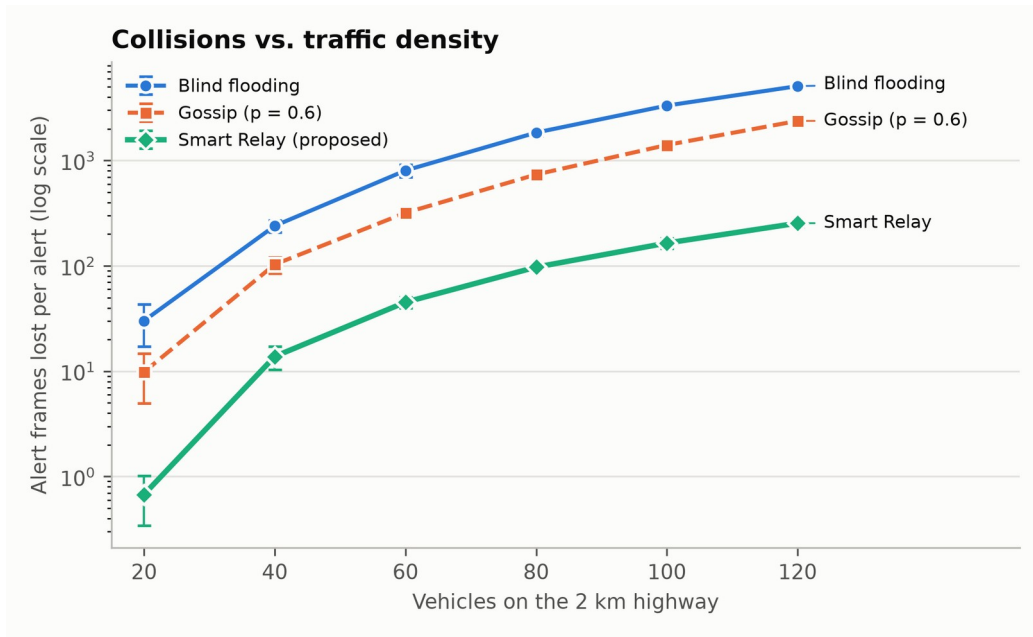


Figure 9.10: Alert frames lost at receivers (collisions), logarithmic scale

Collisions. The number of alert frames lost at receivers (Figure 9.10, logarithmic scale) is about 95 % lower with Smart Relay at every density from 40 vehicles up: 97 against 1,841 per alert at 80 vehicles and 254 against 5,084 at 120 vehicles. This is the direct cause of Smart Relay's higher delivery ratio: with fewer simultaneous transmissions, the relays that do happen are received.

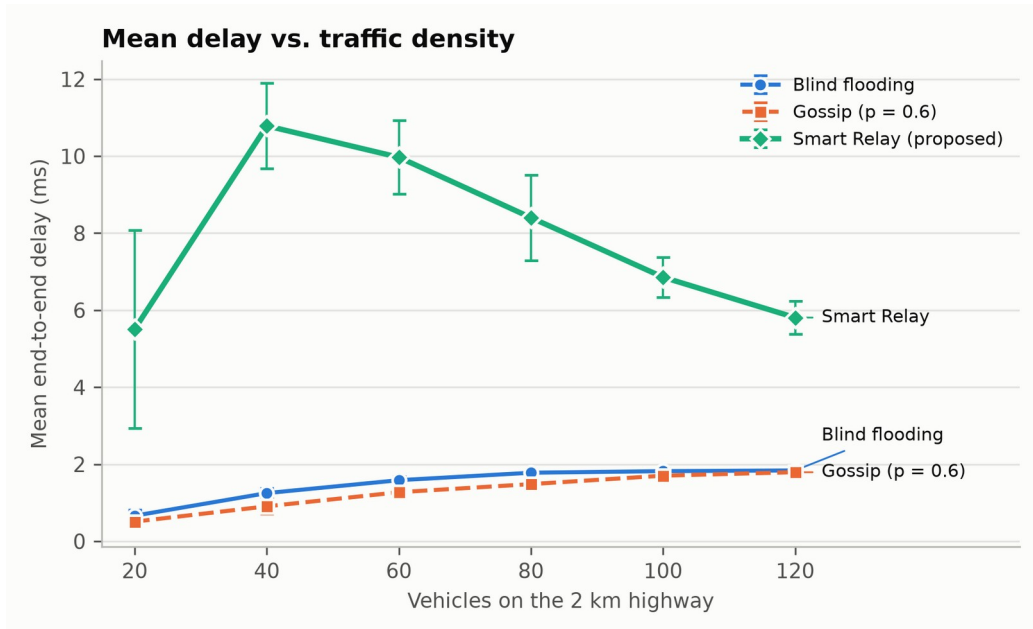


Figure 9.11: Mean end-to-end delay against traffic density

Delay. Flooding and gossip relay immediately, so their mean delay is low, 0.5 to 1.8 ms (Figure 9.11). Smart Relay's waiting slots add delay: its mean is 5.5 to 10.8 ms and its 95th percentile is 15 to 30 ms (Figure 9.12). Two observations matter. First, all values are

an order of magnitude below the 100 ms commonly required for safety warnings. Second, Smart Relay's delay falls as density rises, from 10.8 ms at 40 vehicles to 5.8 ms at 120, because a denser road more often has a vehicle near the edge of the sender's range, which gets slot 0 and relays at once.

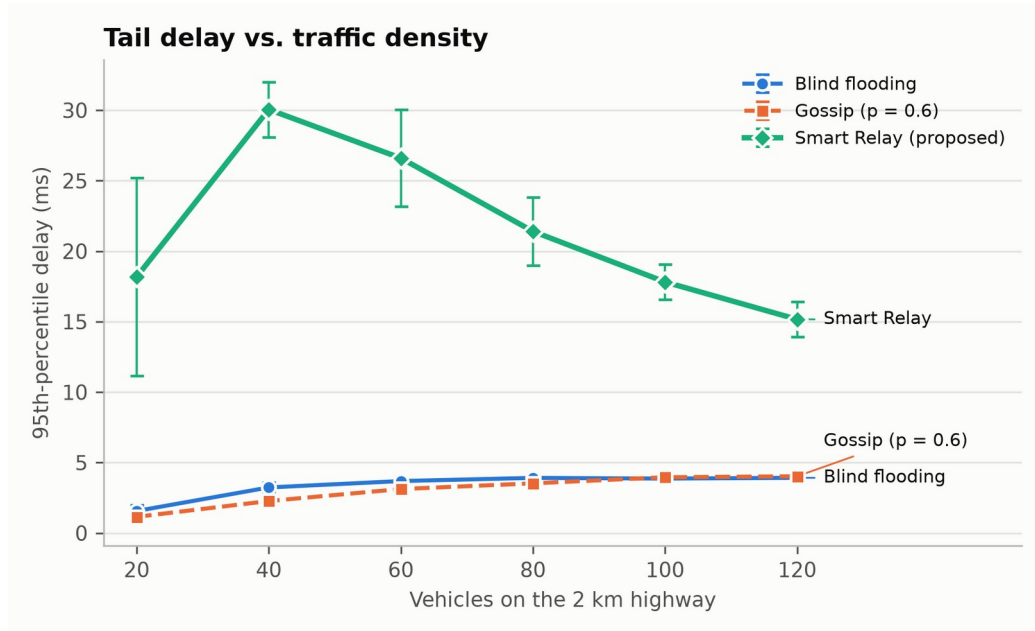


Figure 9.12: 95th-percentile end-to-end delay

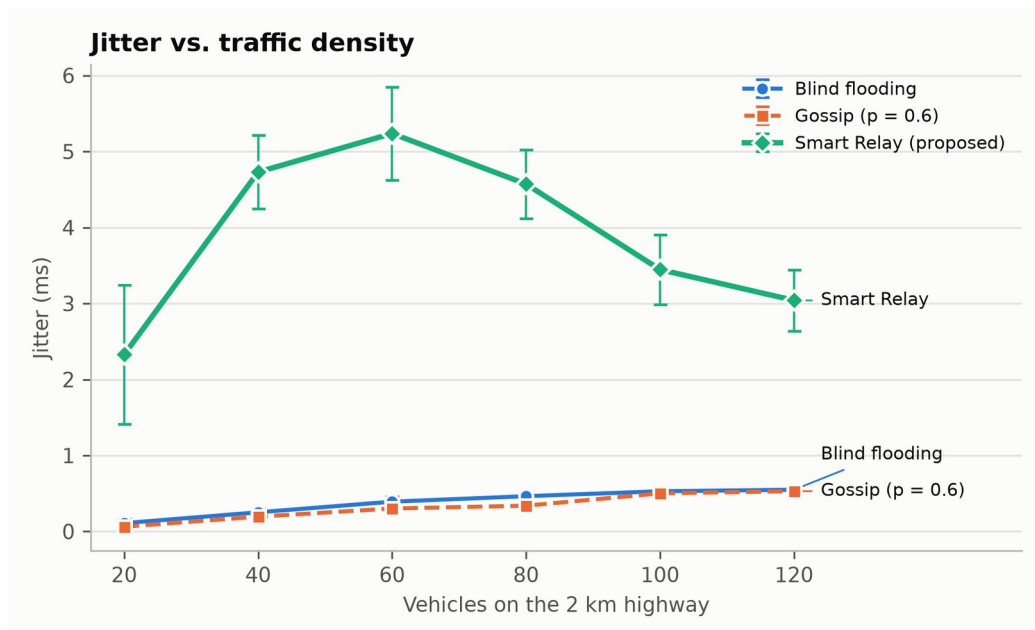


Figure 9.13: Jitter of the end-to-end delay between consecutive alerts

Jitter. Jitter (Figure 9.13) follows the delay: 0.1 to 0.6 ms for flooding and gossip, 2.3 to 5.2 ms for Smart Relay, again falling at higher density. For a warning that is repeated every second, a few milliseconds of variation has no practical effect.

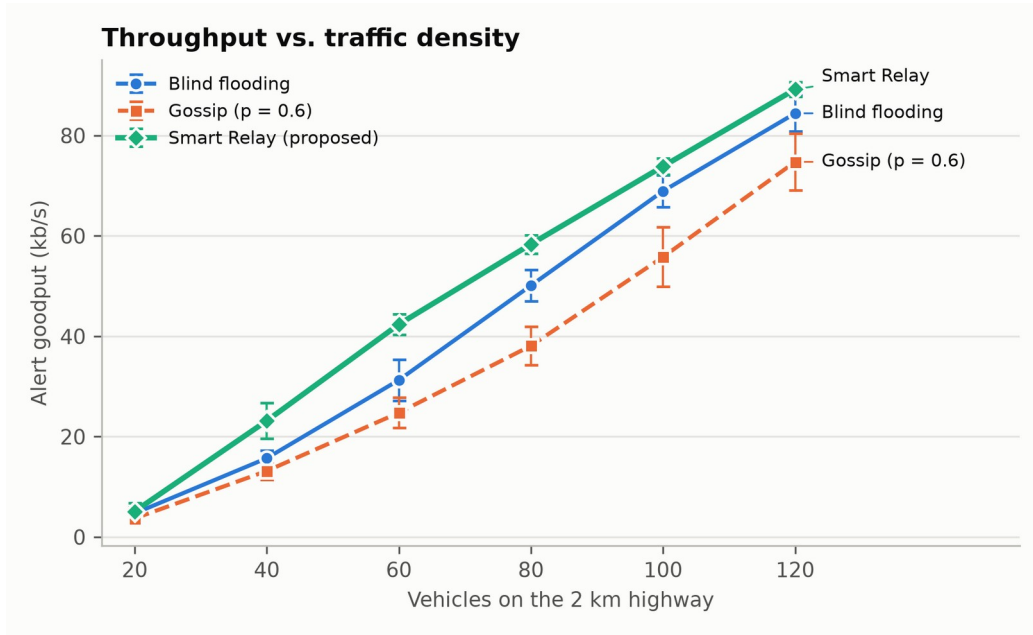


Figure 9.14: Alert goodput (throughput of useful alert data delivered)

Throughput. Goodput, the rate of useful alert data delivered to vehicles in the region of interest (Figure 9.14), is highest for Smart Relay at every density, for example 58.4 kb/s against 50.1 kb/s for flooding at 80 vehicles, because more alerts arrive although fewer are sent.

9.4 Validation of the Proposed Approach

The results were checked in several ways:

- **Analytical delay.** The measured one-hop delay of 0.34 ms in 802.11p matches the 338 μ s computed in Equation (7.2), and the 1.2 ms in version 1 matches the 802.11b calculation.
- **Hop counts.** With a nominal range of about 300 m, reaching vehicles 1.6 km from the crash needs at least six hops; with 40 or more vehicles the observed mean is 3.4 to 4.9 hops and the maximum 9 to 16 hops, which is consistent with this.
- **Fair comparison.** All schemes ran with identical traffic and channel randomness for each seed, so differences come only from the relay rule.
- **Statistical confidence.** Each point is the mean of ten seeds; the 95 % confidence intervals shown in the graphs do not overlap between Smart Relay and flooding for delivery ratio from 40 vehicles upwards, or for transmissions and lost frames from 60 vehicles upwards.

- **Consistency with the literature.** Flooding's collapse under load matches the broadcast storm analysis in [1] and [2], and gossip's all-or-nothing behaviour matches [7].

9.5 Observations

- **Packet delivery ratio:** Smart Relay 78.4 % to 99.3 % from 40 vehicles upwards, against 53.3 % to 93.9 % for flooding and 44.8 % to 83.2 % for gossip.
- **Delay:** Smart Relay 5.5 to 10.8 ms (mean), flooding 1.3 to 1.8 ms; all far below 100 ms.
- **Throughput:** Smart Relay has the highest goodput at every density.
- **Jitter:** Smart Relay 2.3 to 5.2 ms, flooding below 0.6 ms.
- **Overhead:** Smart Relay needs 17 % to 67 % fewer transmissions than flooding and loses about 95 % fewer frames.
- Gossip with $p = 0.6$ is the worst scheme on a highway: it saves transmissions but the alert often dies out.

CHAPTER 10

CONCLUSION

This work implemented a complete emergency alert application for vehicular networks in ns-3.45 using UDP socket programming, and proposed Smart Relay, an application-layer relaying scheme that combines distance-based contention with direction-aware suppression. The application, a custom 40-byte alert header, three relaying schemes, a realistic IEEE 802.11p highway scenario and a measurement framework were built and evaluated over 180 simulation runs.

Smart Relay delivered 96.0 % to 99.3 % of alerts to the vehicles on the highway at 60 to 120 vehicles, compared with 71.0 % to 93.9 % for blind flooding and 56.4 % to 83.2 % for gossip. It needed up to 67 % fewer transmissions than flooding and lost about 95 % fewer frames to collisions, and it had the highest goodput at every density. The price is a higher delay: 5.5 to 10.8 ms on average and about 30 ms at most at the 95th percentile, against about 2 ms for flooding. This is well within the 100 ms budget of safety applications, and it falls as traffic becomes denser.

The work addresses the selected research gap: it shows that a relay rule that needs no MAC changes, no handshakes and no control messages, only the sender's position carried in the alert header, removes most of the broadcast storm on a standard 802.11p stack, under fading and background load and with periodically repeated alerts.

Limitations. The highway is straight and uses a constant-speed mobility model with scripted braking rather than a traffic simulator such as SUMO. Vehicle positions in the header are assumed to be exact (as from GPS). The scheme's parameters (R , N , τ) were fixed rather than adapted to density. When the road is too sparse for any multi-hop path, as at 20 vehicles, no scheme can deliver the alert. Security is not addressed; a forged alert would be relayed like a real one.

Future work. The scheme can be extended with density-adaptive slot lengths, a store-carry-forward mode for sparse traffic, realistic urban maps and traffic from SUMO, authentication of alerts with digital signatures, and a comparison with C-V2X sidelink in ns-3.

CHAPTER 11

REFERENCES

- [1] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in Proc. 5th ACM/IEEE MobiCom '99, Seattle, WA, USA, 1999, pp. 151–162, doi: 10.1145/313451.313525.
- [2] N. Wisitpongphan, O. K. Tonguz, J. S. Parikh, P. Mudalige, F. Bai, and V. Sadekar, "Broadcast storm mitigation techniques in vehicular ad hoc networks," IEEE Wireless Commun., vol. 14, no. 6, pp. 84–94, Dec. 2007, doi: 10.1109/MWC.2007.4407231.
- [3] G. Korkmaz, E. Ekici, F. Özgüner, and Ü. Özgüner, "Urban multi-hop broadcast protocol for inter-vehicle communication systems," in Proc. 1st ACM VANET '04, Philadelphia, PA, USA, 2004, pp. 76–85, doi: 10.1145/1023875.1023887.
- [4] E. Fasolo, A. Zanella, and M. Zorzi, "An effective broadcast scheme for alert message propagation in vehicular ad hoc networks," in Proc. IEEE ICC, Istanbul, Turkey, 2006, pp. 3960–3965, doi: 10.1109/ICC.2006.255700.
- [5] O. K. Tonguz, N. Wisitpongphan, and F. Bai, "DV-CAST: A distributed vehicular broadcast protocol for vehicular ad hoc networks," IEEE Wireless Commun., vol. 17, no. 2, pp. 47–57, Apr. 2010, doi: 10.1109/MWC.2010.5450660.
- [6] H. Füßler, J. Widmer, M. Käsemann, M. Mauve, and H. Hartenstein, "Contention-based forwarding for mobile ad hoc networks," Ad Hoc Netw., vol. 1, no. 4, pp. 351–369, Nov. 2003, doi: 10.1016/S1570-8705(03)00038-6.
- [7] Z. J. Haas, J. Y. Halpern, and L. Li, "Gossip-based ad hoc routing," IEEE/ACM Trans. Netw., vol. 14, no. 3, pp. 479–491, Jun. 2006, doi: 10.1109/TNET.2006.876186.
- [8] X. Zeng, M. Yu, and D. Wang, "A new probabilistic multi-hop broadcast protocol for vehicular networks," IEEE Trans. Veh. Technol., vol. 67, no. 12, pp. 12165–12176, Dec. 2018, doi: 10.1109/TVT.2018.2872998.
- [9] D. Jiang and L. Delgrossi, "IEEE 802.11p: Towards an international standard for wireless access in vehicular environments," in Proc. IEEE VTC Spring, Singapore, 2008, pp. 2036–2040, doi: 10.1109/VETECS.2008.458.
- [10] M. Torrent-Moreno, D. Jiang, and H. Hartenstein, "Broadcast reception rates and effects of priority access in 802.11-based vehicular ad-hoc networks," in Proc. 1st ACM VANET '04, Philadelphia, PA, USA, 2004, pp. 10–18, doi: 10.1145/1023875.1023878.
- [11] H. Hartenstein and K. P. Laberteaux, "A tutorial survey on vehicular ad hoc networks," IEEE Commun. Mag., vol. 46, no. 6, pp. 164–171, Jun. 2008, doi: 10.1109/MCOM.2008.4539481.

- [12] S. Latif, S. Mahfooz, B. Jan, N. Ahmad, Y. Cao, and M. Asif, "A comparative study of scenario-driven multi-hop broadcast protocols for VANETs," *Veh. Commun.*, vol. 12, pp. 88–109, Apr. 2018, doi: 10.1016/j.vehcom.2018.01.009.
- [13] S. S. Shah, A. W. Malik, A. U. Rahman, S. Iqbal, and S. U. Khan, "Time barrier-based emergency message dissemination in vehicular ad-hoc networks," *IEEE Access*, vol. 7, pp. 16494–16503, 2019, doi: 10.1109/ACCESS.2019.2895114.
- [14] W. Benrhaïem, A. Hafid, and P. K. Sahu, "Reliable emergency message dissemination scheme for urban vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1154–1166, Mar. 2020, doi: 10.1109/TITS.2019.2902850.
- [15] D. Cao, B. Zheng, B. Ji, Z. Lei, and C. Feng, "A robust distance-based relay selection for message dissemination in vehicular network," *Wireless Netw.*, vol. 26, no. 3, pp. 1755–1771, 2020, doi: 10.1007/s11276-018-1863-4.
- [16] F. Raviglione, C. M. Risma Carletti, M. Malinverno, C. Casetti, and C. F. Chiasserini, "ms-van3t: An integrated multi-stack framework for virtual validation of V2X communication and services," *Comput. Commun.*, vol. 217, pp. 70–86, Mar. 2024, doi: 10.1016/j.comcom.2024.01.022.
- [17] W. R. Stevens, B. Fenner, and A. M. Rudoff, *UNIX Network Programming, Volume 1: The Sockets Networking API*, 3rd ed. Boston, MA, USA: Addison-Wesley, 2003.
- [18] ns-3 Project, "ns-3 Network Simulator, release 3.45: Manual, Model Library and API documentation." [Online]. Available: <https://www.nsnam.org/documentation/>