Authors' version (postprint)

Evaluating the Safety Impact of Network Disturbances for Remote Driving with Simulation-Based Human-in-the-Loop Testing

Shrishti Trivedi and Fredrik Warg

Published/presented in:

1st International Workshop on Verification & Validation of Dependable Cyber-Physical Systems (VERDI 2023). Co-located with DSN 2023.

DOI: 10.1109/DSN-W58399.2023.00059

© 2023 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Postprint – Appears in *1st Intl. Workshop on Verification & Validation of Dependable Cyber-Physical Systems (VERDI 2023)*

Evaluating the Safety Impact of Network Disturbances for Remote Driving with Simulation-Based Human-in-the-Loop Testing

Shrishti Trivedi[§]
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
Stockholm, Sweden
shrishti@kth.se

Fredrik Warg
Dependable Transport Systems
RISE Research Institutes of Sweden
Borås, Sweden
0000-0003-4069-6252

***Abstract*—One vital safety aspect of advanced vehicle features is ensuring that the interaction with human users will not cause accidents. For remote driving, the human operator is physically removed from the vehicle, instead controlling it from a remote control station over a wireless network. This work presents a methodology to inject network disturbances into this communication and analyse the effects on vehicle manoeuvrability. A driving simulator, CARLA, was connected to a driving station to allow human-in-the-loop testing. NETEM was used to inject faults to emulate network disturbances. Time-To-Collison (TTC) and Steering Reversal Rate (SRR) were used as the main metrics to assess manoeuvrability. Clear negative effects on the ability to safely control the vehicle were observed on both TTC and SRR for 5% packet loss, and collision analysis shows that 50ms communication delay and 5% packet loss resulted in crashes for our test setup. The presented methodology can be used as part of a safety evaluation or in the design loop of remote driving or remote assistance vehicle features.**

***Index Terms*—Automotive safety, remote driving, tele-operated driving, remote assistance, human-in-the-loop testing, simulation, fault injection.**

I. Introduction

The use of connected services, automation, and electrification in the transportation domain is increasing rapidly. One feature made possible by recent technological advancements, in particular ubiquitous high-speed communication, is Remote Driving (RD) or Tele-operated driving. With this technology, a driver can partially or fully perform the driving task using a remote control station [4] instead of being physically present in the vehicle. A Remote Driving System (RDS) have several use cases ranging from fleet operations and shuttles to dedicated areas like seaports and mining [1].

RDS is a feedback-control system where a video feed is sent from the vehicle to the remote control station, and driving commands from the control station to the vehicle. Any disturbance in communication like delay or blurred video feed can impact the remote driver's ability to control the vehicle. If this impact is significant, the consequence of lost control can be a collision causing material damages or, if the vehicle is used in environments with manual vehicles or pedestrians, even result in serious injuries or fatalities. Hence, RD will require appropriate safety measures to handle such disturbances. To investigate which safety measures are adequate, e.g., how they should be designed and when they need to intervene, and to be able to validate that the implemented measures actually perform as expected, comprehensive testing is needed.

This work presents a methodology to use simulation-based human-in-the-loop testing for analysing the effects of network disturbances for RD. The methodology is aimed at complementing tests using real vehicles, by providing an environment enabling faster feedback and testing of a larger number of scenarios due to the flexibility of a simulated environment. The CAR Learning to Act (CARLA) simulator [2] is used as the vehicle model, and connected to a driving station. The NETEM [3] tool is used to introduce network disturbances, like communication delays and packet loss. To evaluate the methodology, 11 tests were performed in a test setup without any safety measures to counteract network disturbances. The test subjects were asked to drive through different test scenarios both with and without faults injected. These runs were analysed to understand the effect on the manoeuvrability of the vehicle. Two main safety metrics were used - TTC and SRR. Significant negative effects on manoeuvrability were observed on both metrics for 5% packet loss, and collision analysis shows that tests with 50ms communication delay or 5% packet loss also resulted in crashes in this test setup.

Section II provides a brief background on RD, safety evaluation, and network fault injection. The main contributions of the paper, methodology, experimental design, and results & analysis, are described in sections III, V, and VI respectively. Section IV points out relevant research areas that were out of scope of this work. Section VII revisits the research questions and finally, section VIII concludes the work.

This work was partly supported by VALU3S project, which has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey.

[§]The work was performed during the author's master's thesis project at RISE Research Institutes of Sweden.

II. Background & Related Work

A. Remote Driving Evaluation

Work in paper [4] presents a RD setup for the evaluation of 4G and 5G networks. The setup in the control station includes pedals, a steering wheel and three monitors. For the network setup, Radio Access Networks (RAN) sharing was used for 4G, and dedicated antennas for 5G. The authors mention that it is challenging to alter the network to produce varying network efforts like traffic congestion on the network, distance from the antenna, etc. in such a setup. For that reason, virtual testing is also performed, where a simulated environment replace the real cars, and the network is emulated to produce different network conditions. The two network conditions tested are latency and packet loss for both straight-line and slalom tests. The network conditions are tested both uni-directionally and bi-directionally.

In [5], the authors use a scaled-down prototype instead of a real vehicle to keep the experiments reasonably realistic yet inexpensive. In this setup, three cameras were used to cover the front and side views. A smartphone was used to connect the vehicle to the remote station and NETEM was used for introducing the delays in the network. For the remote control station, three monitors (one for each camera feed), a Logitech G27 racing wheel, and pedals were used.

The work most similar to ours, TELECARLA [6], presents a human-in-the-loop RD simulator solution using CARLA. The setup is also similar to other works in using a gaming seat, gaming wheel and pedals, and three display monitors. The vehicle subsystem, in this work, is replaced by the driving simulator CARLA, and the work uses Robot Operating System (ROS) as middleware to establish a connection between the CARLA client and the CARLA server. The authors suggest that using ROS is beneficial to compress data transmission and provides a framework that is more modular and scalable. Our work uses a somewhat simpler setup without ROS.

B. HMI Safety Evaluation

The standard ISO/PAS 21448 or SOTIF [7] focuses on functional inefficiencies, i.e., hazards due to for instance unforeseen operating conditions or gaps in the specification. The scope also includes inadequate design or implementation of the Human-Machine Interface (HMI), which is highly relevant for an RDS. Thus, the methodology presented in this paper could be used as part of the verification and validation activities in the SOTIF lifecycle, e.g., together with an HMI safety analysis method such as the one presented in [8].

Jahangirova et al. [9], performed an extensive study to find metrics that can be utilised for human drivers' driving quality assessment. They also analysed the correlation of 26 of these metrics with safety metrics for Automated Driving Systems (ADS). According to the authors, statistical metrics like mean, max, and standard deviation for brake, speed, acceleration as well as the frequency of crashes or braking, headway time, and SRR can be considered useful for both ADS and human driving. The Society of Automotive Engineers (SAE) report *'AVSC Best Practice for Metrics and Methods for Assessing Safety Performance of Automated Driving Systems'* [10] mentions crash severity and frequency, maintaining safety envelope, acceleration, and following traffic regulations as safety metrics. Review paper [11] highlights traditional metrics like TTC, Post Encroachment Time (PET), and Time Exposed TTC (TET). Longitudinal and lateral positions, accelerations, speeds, lane offset, and headway are considered for measuring human driving behaviour. SAE J2944 [12] details how to measure driving performance for longitudinal and lateral control based on the driver's pedal responses and vehicle measurements. The metrics include Headway Time (HT), TTC, TET, SRR, and lane departure. In [13] the authors mention that different countries have different safety and legal regulations on the minimum distance a car should maintain from others, e.g., [14] specifies European regulations, for passenger cars two seconds. We have used TTC and SRR, along with collision analysis, for our experiments since these are common metrics for evaluating driver behaviour and assistance systems, and also relatively simple to calculate. However, an analysis of the suitability of different metrics for RD evaluation is beyond the scope of this paper.

C. Network Fault Injection

Network emulation allows for performing tests of realistic network conditions in a controlled manner. Advantages include no changes required in network infrastructure, and an ability to easily test different network conditions. A survey by Nussbaum and Richard [15] compares network emulators. They can be divided into two types – virtual network emulators and network link emulators. The former emulates a group of computers on a network whereas the latter emulates the network condition on the ingress or egress traffic from the network interface. Some examples of virtual network emulators are Modelnet, eWAN, and Emulab, and examples of network link emulators are DummyNet, NISTNet, TC/NETEM, and WANem. Linux traffic control manipulates the queuing systems responsible for receiving and transmitting packets on the router. NETEM is based on Linux traffic control and can be used to emulate a wide area network, injecting variable delay faults, loss, duplication and re-ordering, fixed or non-variable delay, packet loss, packet duplication, packet corruption, packet re-ordering, and rate control for bandwidth [3], and is the tool we have used in our setup.

III. Methodology

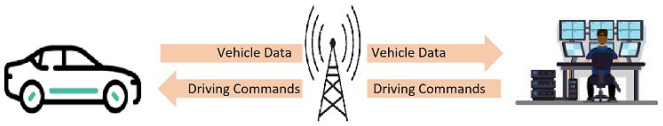
The primary question, how the communication disturbances for RD or remote assistance affect the manoeuvrability of the vehicle, is broad and includes multiple aspects. The two more specific research questions tackled in this paper are:

1) How to efficiently perform and evaluate driving tests for an RDS under network disturbances?
2) How to use such driving tests during the system design phase to improve the safety of the RDS?

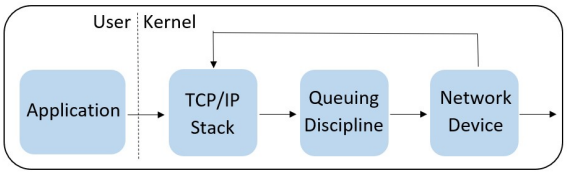
A. RDS

According to [1], a typical RDS can be divided into four subsystems. Fig. 1a shows a basic RDS. The subsystems are:

- Vehicle subsystem: Manages many tasks including providing sensor information to the operator to "perceive" the environment and also communication delay and Quality of Service (QoS) information that should be considered while performing driving commands.
- Operator subsystem: (i) Uses sensor data to produce as realistic an environment as possible for the operator, i.e., focuses on user experience. (ii) Receives driving commands from the remote operator and must transmit them to the vehicle subsystem within the time constraints.
- Infrastructure subsystem (optional): Improves the environment perception by providing more sensor data from additional sources than the vehicle subsystem.
- Communication network subsystem: Supervises and conducts reliable and low-latency communication of the sensor data and driving commands.



(a) RDS subsystems.



(b) NETEM architecture.

Fig. 1: Simulation methodology.

B. Simulator-in-Loop: CARLA

Using a real car for testing is time-consuming and expensive. In addition, it is challenging to produce scenarios that are dependent on environmental conditions. Simulation-based testing, on the other hand, is cheaper to develop, safer, reusable, reproducible, and has the possibility to run batch automated tests [16]. Several driving simulators are available, but the commercial options are too expensive for studies such as ours, and others, which are freely available, have limitations in terms of road users or road networks [2]. CARLA [2], [17] is an open-source driving simulator often used for the ADS testing due to good scalability, flexibility, and features like traffic generation. Simulation-based testing thus seems suitable for efficiency, which was part of our first research question. For our purposes, however, including a human-in-the-loop is necessary as the objective is to investigate the impact of network disturbances on the remote driver. Therefore, we use CARLA connected to a remote control station.

CARLA has three basic components, the simulation engine, the simulator environment, and the sensor suite. The engine is based on a server-client architecture with communication over Transmission Control Protocol (TCP). The server handles rendering physics for actors, running simulations, and establishing communication with the client. The client is responsible for controlling the actor and agent's logic. This is done by sending commands (steer, reverse, brake, and accelerate) and meta-commands that affect the server's behaviour such as weather, sensor properties, and road users. The simulator environment consists of static objects such as vegetation and buildings, and dynamic objects like pedestrians and vehicles. The sensor suite offers several sensors including RGB, LIDAR, and collision sensors, for collecting data. Other measurements include recording vehicle dynamics like velocity, location, acceleration, and orientation. We use some of these sensors to evaluate driving behaviour.

C. Network Fault Injection: NETEM

NETEM, an improvement of Linux Traffic Control (TC), is used to emulate network conditions to simulate transmission processes, and can thus directly emulate different types of faults without a need to replicate the potential root causes of these faults (such as bad environmental conditions for a wireless connection). There are three main parts of TC - queuing discipline, class, and classifier. The queuing discipline can control the transmission speed of the network. The application sends the data packets to the TCP/Internet Protocol (IP) stack. Further, these data packets are transferred to the queuing discipline and further sent out to the network. With NETEM, it becomes simple to modify the rules for these queuing disciplines, allowing modification of several network conditions including delay, packet loss, packet corruption, packet duplication, and packet reordering. The architecture is shown in Fig. 1b. NETEM configuration includes which network device will be affected and which condition will be altered, e.g., `delay 50ms`.

IV. Delimitations

This work focuses on understanding how to perform human-in-the-loop testing for a RDS and the potential impact on road safety during the presence of network disturbances. The scope is hence limited and the following are not considered:

- There can be several sources of origin of the network disturbances including but not limited to poor network coverage, bad weather, cybersecurity attacks, or traffic congestion. This work is limited to evaluating how road safety will be affected in the presence of network disturbances irrespective of the cause of the fault.
- The setup is experimental and still in the development phase. The validity is not evaluated, hence no concrete conclusions about road safety can be drawn from this specific setup. The aim is to demonstrate the methodology.
- The driving behaviour of the test subject has not been a part of the study. It is important to note that this limits the usefulness of the safety metrics used for comparison.

V. Experimental Design

A. RDS Setup

The driving station consists of a gaming seat, steering wheel, pedals, and a curved screen (see Fig. 2a). CARLA, 0.9.12 packaged version was used as the driving simulator. The client (remote station) and server (vehicle subsystem) were running on the same computer. By using only a loopback network interface for the fault injection no external network disturbances are added to the experiments. The video frame rate of the simulator was in the range of 25 to 30 frames per second. Technical specifications are shown in Table I.

TABLE I: Technical Specifications for Driving Station.

CPU and RAM	Intel Core i7-12700K (12-core), 16 Gb RAM
Monitor	34" Samsung WQHD (3440x1440) curved
Input device	Logitech G27 steering wheel and pedals
GPU	NVIDIA GeForce RTX 3080, 10 Gb
Operating system	Ubuntu 18.04
NVIDIA driver	470.103.01

B. Driving Scenarios

The remote operator should not only drive safely but also follow applicable traffic rules. Hence, driving scenarios were designed based on some of the driving proficiency requirements for a Swedish driving license [18]. The remote driver must be able to safely manoeuvre the vehicle using the pedals and steering on both straight and curved roads, while maintaining a safe distance to other road users. The defined scenarios include following a vehicle, lane change operation due to a stationary vehicle, and overtake. The Operational Domain (OD) for the tests was Town 5 in the CARLA package (Fig. 2c) which includes a highway and multi-lane road network, day and night time conditions, and presence of one dynamic and a few static road users. Two false test cases (cyclists where the driver might think intervention would be necessary, but actually wasn't) were also present. A front view of a scenario as seen by the driver is shown in Fig. 2b.

C. Fault Model

Defining a fault model means selecting the fault type, value, location, and duration. The initial candidates included delay, packet loss, corruption, and duplication. The last two did not show any clear visual or operational effect on the simulator and hence were discarded. The user experience for delay faults was indeed a delay between the command and its effect in the video, while for packet loss, it was rather that of certain frames being skipped. However, a clear visual difference between these faults was often difficult to spot. 5ms, 25ms and 50ms delays and 2% and 5% packet loss faults were finally selected based on initial testing, with the purpose of exploring the limits of manoeuvrability, i.e., they show some effect but do not make it impossible to drive. The fault location was points of interest while following a vehicle, and when performing lane change operations between stationary vehicles. The fault injection was done randomly and with a duration dependent on the situation, i.e., if a 5ms delay was injected for one test subject, a 5% packet loss might have been injected in the same scenario for another subject. This was done to get insights into how different faults affected driving on average; testing the same scenario/fault combination for all subjects would make it difficult to distinguish the effect of the fault from the effect of the scenario itself.

D. CARLA Fault Injection

The server renders the simulation and sends video to the client (driving station) through the virtual network interface. Steering, throttling, and braking commands are sent back to the server. Using NETEM, the faults are injected by adding or deleting rules for the interface, altering the outgoing traffic. As both server and client are running on the localhost, both video data, commands, and meta-commands become outgoing traffic; hence, the fault injection is bidirectional, see Fig. 3.

E. Test Process

The tests were conducted in three steps and each test was 20-30 minutes long. The test subjects were all employees at RISE[1]. All subjects have a driving license.

1) Step 1: Training Time: All the test subjects were given a minimum of three minutes and a maximum of five minutes to drive around freely in an empty town in CARLA. This was done to get familiarised with the driving station, especially the sensitivity of the steering wheel and the pedals.

2) Step 2: Testing Time: This part consists of two runs, a golden run (when no faults were injected) and a faulty run (when faults were injected). The reason to do a golden run was to have a baseline reference of driving behaviour for each test subject. The test subjects were given instructions during the test regarding the turns they should make to manoeuvre around different test scenarios. The faults were injected whenever the subject came across a situation of interest.

3) Step 3: Short Questionnaire: After the driving tests, the test subjects were asked the following questions:

1) Do you have much experience playing video games?
2) Have you played any car racing games, specifically?
3) Do you have any professional and/or personal experience with the driving station beforehand?
4) How would you describe the Quality of Experience (QoE) of the second run as compared to the first run?
5) To what extent do you agree that virtual testing is useful for testing purposes?
6) Did you feel any difference in the faults injected?

The purpose of the first three questions was to see whether previous experience with this type of setup has a big impact on the results, and the last three questions were asked to obtain the test subjects' own assessment of the tests to complement the road safety metrics.

[1]RISE Research Institutes of Sweden, https://ri.se/en

(a) Driving station (b) View seen by remote driver (c) CARLA map used for tests

Fig. 2: Experimental setup with driving station and CARLA.

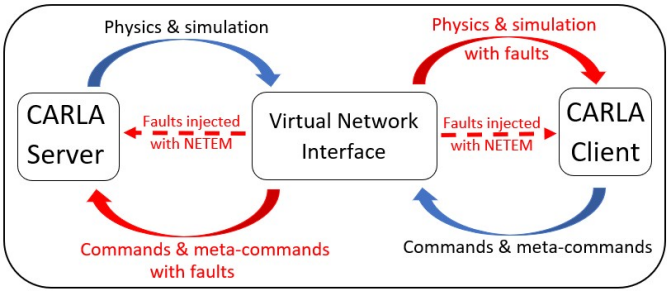


Fig. 3: Working architecture of simulation-based testing with network disturbances

F. Data Logging

The CARLA sensor suite was used to log data in both the golden run and faulty run, including these parameters:

- Collisions: timestamp, frame, collision actors.
- Lane invasions: timestamp, frame, lane that is invaded.
- Ego vehicle: timestamp, x, y, z, v_x, v_y, v_z, a_x, a_y, a_z, throttle, steer, brake.
- Other vehicle: actor, timestamp, distance from ego vehicle, x, y, z, $v_x, v_y, v_z, a_x, a_y, a_z$, throttle, steer, brake.
- Fault injection: timestamp, fault type, value, added/deleted.

G. Road Safety Metrics

1) TTC: This metric calculates time to collision if two vehicles continue to move in the same direction with their current velocities, and is thus relevant especially for longitudinal control. Larger TTC indicate a lower risk of collisions.

$$TTC = \frac{X_L - X_F}{v_F - v_L}$$

where X_L and v_F is the leading vehicle position and speed, and X_F, v_F for the following vehicle. A threshold value can be used, indicating a violation if $0 <$ TTC $<$ threshold.

2) SRR: To evaluate the lateral control performance for curved roads or slalom, SRR can be considered. SRR refers to rotation of the steering wheel for greater than some threshold angle in one direction and then in the other direction within a specific time frame. An algorithm to calculate SRR is presented in [12]. The basic idea is to apply a low-pass filter to remove any noise in the steering signal, find the stationary points, and then count the reversals.

Answers from the questionnaire can be used to correlate the driving performance with a RDS setup. For example, if experience with video games positively correlates with better performance even in the presence of faults, it could be used to influence the remote driver training.

VI. Results & Analysis

This section presents results from the experiments, with a focus on the key observations[2].

A. Data Processing

Data was originally collected from 12 test subjects, denoted T1-T12 in the tables. However, some data had to be excluded due to mistakes in the collection phase:

- T7 had to be disregarded as it was revealed that the subject is used to left-handed driving, which unduly affected the ability to drive in our (right-hand) scenarios[3].
- Some data were not recorded properly due to technical issues. Steering data for T3 for No Fault Injected (NFI) and T8, T10, and T12 for Fault Injected (FI) runs. Hence the SRR are missing for these tests. The velocity of the dynamic vehicle was missing for T1, T2, T3, and T4 for both NFI and FI runs. Because of this, TTC analysis of these tests could not be performed.

[2]Note that the quantitative findings are subjective only to the given RDS configuration; the aim of the work is not to find fault limits for RD in general.

[3]The remaining tests have not been renamed to keep the numbering intact between this paper, the raw data, and the master's thesis.

B. Faults Injected

Five faults, three delays (5ms, 25ms, 50ms), and two packet loss frequencies (2% and 5%) were injected. Table II shows the number of faults of different type injected in each test[4].

TABLE II: Summary for Faults Injected

Frequency of Faults						
Test	Delay			Packet Loss		Total
	5ms	25ms	50ms	2%	5%	
T1	3	1	2	3	1	10
T2	3	2	2	2	3	12
T3	3	4	1	2	3	13
T4	1	4	1	4	1	11
T5	2	2	2	2	2	10
T6	2	3	2	2	3	12
T8	1	4	3	2	3	13
T9	1	2	3	3	3	12
T10	1	2	3	4	4	14
T11	2	3	3	2	3	13
T12	1	3	2	5	3	14
Total	20	30	24	31	29	134

C. Driving performance evaluation: TTC

For calculating the TTC, only intervals with relative distance between the lead and the ego vehicles $\leq$ 100m were included; due to relatively low speeds, longer distances always result in a high TTC. Results are shown in Table III. Column NFI shows the TTC values for the golden run. "-" means that either the fault was not injected or the distance was always more than 100m during fault injection. Key observations:

- The average and maximum TTC are lower for most of the tests compared to the respective TTC in the golden run. This indicates a higher risk for collisions. The minimum TTC increased in most of the faulty run situations. This indicates that the test subjects were driving more cautiously in presence of network disturbances.
- According to [13], TTC $>$ 6s is not considered dangerous. With this threshold, a packet loss of 5% results in a safety violation whereas a 5ms delay does not cause significant violations. T6 is considered an outlier with low TTC throughout the tests. The remaining faults could be considered potentially dangerous, as on average, these faults result in TTC closer to the threshold in several tests.

D. Driving performance evaluation: SRR

According to SAE J2944, [12], smaller steering wheel corrections happen when the human driver is more attentive. This means that higher SRR signify that the drivers were distracted or disturbed, which in this case is assumed to be highly influenced by the network disturbances. Table IV shows the results from SRR where NFI and FI column depicts average SRR values for the golden run and faulty run during the entire test respectively. The "Avg." column shows the average of the SRR values, for all faults, when the fault was injected. "x" in the table depicts that the data was not recorded. SRR analysis has the following main observations:

- For three tests, (T1, T2, and T9), SRR recorded is higher in all faults compared to when no fault is injected. For tests T5 and T6, it remains higher for 80% of the faults, and for T4 for more than 50% of the faults. Even though the steering data was not recorded for T3 (in the golden run), the results for the faulty run suggest high SRR.
- Average SRR for all three delays are quite similar, which can indicate that delay network disturbances do not have a high impact on latitudinal control.
- Highest SRR is observed for packet loss 5%.
- In the survey, five test subjects (T1, T2, T4, T10, and T11) reported that they felt visual differences between the faults, but the results do not show that these test subjects have higher SRR in general. This is contradictory to the study in [19]. We speculate that this is because the visual differences between the faults are not as noticeable in our experiment. In their study, visual distractions like black boxes are presented on the screen to the test subjects.
- From Fig. 4 it can be seen that the driver took a longer time to navigate through the same scenario, in presence of network faults. For example, the left circle represents a scenario with three vehicles. The test subject took around 19s to perform lane change operation and manoeuvre around the vehicles in the golden run (bottom) whereas 33s in the faulty run (top).

TABLE III: Statistics for TTC (in sec)

Maximum TTC						
Test	NFI	Delay			Packet Loss	
		5ms	25ms	50ms	2%	5%
T5	68.77	21.42	-	36.25	44.69	-
T6	93.22	0.87	0.53	0.72	39.47	0.89
T8	74.56	-	2.97	3.09	-	-
T9	77.48	-	16.58	22.95	23.94	2.69
T10	77.79	-	31.12	29.19	83.72	-
T11	88.51	16.37	-	29.59	13.08	-
T12	55.04	15.06	16.58	34.77	74.66	-
Average TTC						
Test	NFI	Delay			Packet Loss	
		5ms	25ms	50ms	2%	5%
T5	13.31	10.39	-	6.16	8.99	-
T6	11.71	0.71	0.42	0.43	6.80	0.06
T8	14.12	-	2.00	2.06	-	-
T9	17.87	-	12.25	7.41	12.22	2.69
T10	12.23	-	12.30	12.89	14.55	-
T11	17.58	12.32	-	9.26	13.03	-
T12	13.22	11.27	6.87	20.83	25.40	-
Minimum TTC						
Test	NFI	Delay			Packet Loss	
		5ms	25ms	50ms	2%	5%
T5	2.64	6.50	-	0.35	2.31	-
T6	1.65	0.59	0.36	0.32	2.22	0.44
T8	3.81	-	1.12	1.02	-	-
T9	1.14	-	10.85	4.09	7.28	2.69
T10	0	-	6.18	5.67	4.86	-
T11	1.23	10.62	-	4.36	13.02	-
T12	0.85	8.62	3.89	4,44	14.57	-

[4]The authors recognise that the low number of tests and similar experience of test subjects might skew the results. This highlights that although useful, human-in-the-loop tests are time-consuming; more tests and subjects with varied experience would provide more reliable results.

TABLE IV: Statistics for SRR (in revolutions per minute)

Test	NFI	FI	Delay			Packet Loss		Avg
			5ms	25ms	50ms	2%	5%	
T1	4.5	3.9	6.7	6.8	5.2	9.8	8.3	7.36
T2	5.7	6.2	9	8.6	9.3	13.7	11.2	10.36
T3	x	7.6	13.3	11.2	11	6	11.6	10.62
T4	6	4.3	4.2	6.2	5.3	8.8	2.8	5.46
T5	4.2	5.2	2.1	9.3	9	6.5	14.4	8.26
T6	5.4	6	6.4	6.7	10	2.9	8.7	6.94
T8	3.4	x	x	x	x	x	x	x
T9	5.1	6.8	13.5	7.3	8.3	9.6	8	9.34
T10	5.3	x	x	x	x	x	x	x
T11	5.8	4.7	5.4	6.7	3.2	4.4	8.5	5.64
T12	5	x	x	x	x	x	x	x
Avg	5.04	5.58	7.57	7.85	7.66	7.71	9.18	7.59

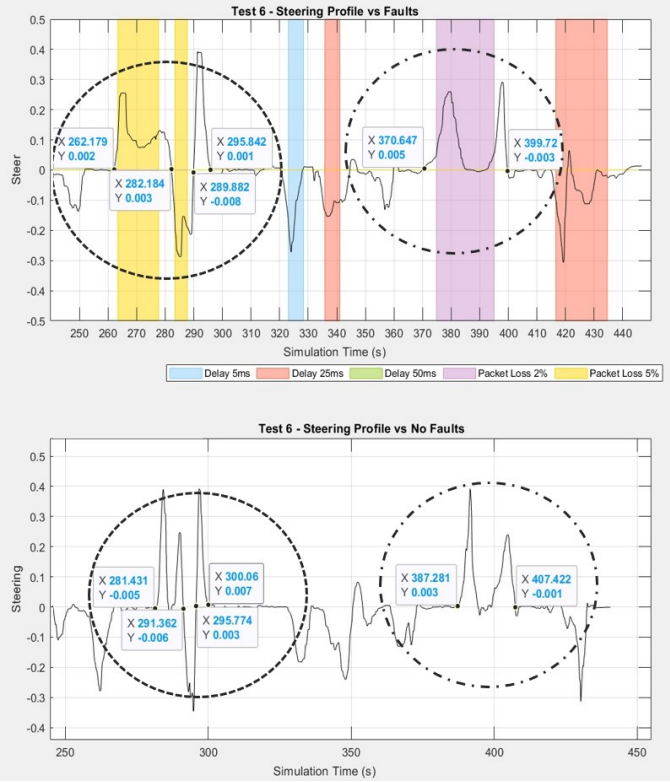


Fig. 4: Results from steering profile

E. Driving performance evaluation: Other Metrics

Some more analysis on velocity, acceleration, and driving profiles (throttle and brake) was done although no concrete trends were found in these cases. The reason behind this could be not injecting the same fault in the same scenario for all test subjects, which made it difficult to spot patterns. Lane invasion data was collected but is not yet analysed. However, some interesting observations were made using collision data analysis. Out of 11 participants, only two collided during the golden run, but eight participants collided in the faulty run. This corroborates the conclusion that network disturbances affect driving. It is also observed that only two types of faults led to crashes, a delay of 50ms, and packet loss of 5%.

F. Answers from Questionnaire

The results from these questions can be used to correlate the driving performance with the faults injected. In this work, only the last question was used, where a contradiction was found (see Section VI-D). Question 3 could be a good basis for correlation, but was not analysed due to limited time. Correlation for gaming experience could not be done since most of the test subjects had previous experience, highlighting a lack of diversity in our group of test subjects. Summary of questionnaire answers:

1) 10 out of 11 test subjects have some previous experience in playing video games but not recently. Only one test subject has recent experience.
2) 9 out of 11 test subjects have explicit experience in playing car-racing games.
3) 6 test subjects reported no prior experience with a driving station. Three of them have used similar setups a few times, and the remaining two only once.
4) The mean QoE is 2.81, with 2 being the minimum and 4 being the maximum score.
5) All test subjects believe virtual testing can be useful.
6) 5 out of 11 test subjects reported visually seeing the difference when the faults were injected.

VII. Discussion

This section revisits the research questions from section III.

1) *How to efficiently perform and evaluate driving tests for an RDS under network disturbances?*
   For efficiency, we suggest simulation-based human-in-the-loop testing. To perform driving tests for a remote-operated vehicle under network disturbances, a driving station needs to be set up with a network fault-injection tool like NETEM. The driving station should include at least one monitor for display, steering wheel and pedals, and either a remote-operated vehicle or driving simulator. For evaluating driving tests, road-safety metrics like TTC and SRR could be considered powerful. This work focuses on a possible methodology to include driver-in-the-loop in the design of RDS. The usage of a questionnaire reinforces and validates the methodology proposed.
2) *How to use such driving tests during the system design phase to improve the safety of the RDS?*
   The results from the driving tests highlight that it is important to understand driving behaviour and consider HMI to understand the effect on road safety holistically. Correlating the driver's prior experience with the RDS setup with their driving performance can also provide some useful insights on road safety in the presence of network disturbances. This highlights the importance of the questionnaire and also a need to understand driving behaviour to validate methodology. The results also signify that there is a need to develop new as well as improve existing safety standards focusing exclusively on RDS for both testing strategies and safety metrics for evaluation.

VIII. Conclusions & Future Work

A typical RDS consists of a remote operator subsystem, a vehicle subsystem, and the communication network between the two subsystems. Including a simulator-in-loop as part of the testing, verification, and validation process not only provides intriguing insights about the system design but also offers advantages compared to testing on a prototype or a real car. The communication network between the remote operator and the vehicle is of paramount importance and any disturbance can result in safety violations as well as accidents, injuries, or even fatalities. This reiterates the need to understand how network disturbances affect the manoeuvrability of the vehicle and affect driving performance.

This work presents a way to analyse the driving behaviour in the presence of network disturbances with simulation-based testing with a human-in-the-loop. For doing this, CARLA and NETEM were used as the driving simulator and network fault-injector tool, and 11 successful tests were performed. Two particular driving scenarios, following a vehicle and lane change using a slalom situation, were considered. The driving performance was analysed based on safety metrics like TTC, SRR, and frequency of collision. Analysing the "following a vehicle" driving scenario indicated a shorter average and maximum TTC during a faulty run, e.g., if 6s is considered as a threshold for a TTC violation then a delay of 5ms never caused a safety violation whereas a packet loss of 5% caused a violation for both completed tests. Results from SRR suggest that out of seven tests, four show higher SRR when faults are injected. Another important observation is that the steering compensation was longer in many cases. Collision analysis also depicted an increase in the number of collisions in the faulty run as compared to the golden run. Overall, the results indicate that the driving performance was affected in presence of network disturbances.

The validity of our results depends on the accuracy of the simulation model, in this case, CARLA, as well as setup and calibration of the driving station. Delays $>$ 100ms and $>$ 200ms made it difficult to drive and stopped the simulator to respond completely, respectively. A packet loss of 1% had no significant effect but a packet loss of 10% made it very difficult to drive. To compare the validity, some fault-injection tests were also performed on a remotely operated model vehicle. It was found that any delays $>$ 20ms resulted in degraded driving but delays $>$ 100ms made it impossible to drive. Similarly, a 7% packet loss created a conscious impact on the ability to drive and a 10% packet loss made it impossible to drive. Although the observations were similar for both environments, suggesting simulation may be a useful complement to real vehicle tests, sufficient comparisons with a real vehicle have not been done, hence the validity of the setup remains an open question. Other potential future work includes a more efficient test plan, a more diverse group of test subjects to be able to compare results for different levels of driving and gaming experience, and also to evaluate more combinations of fault models and different scenarios.

Acknowledgment

This work was performed as part of a larger effort with three related master thesis projects, and topics related to the ongoing research projects VALU3S and SCAT. We would like to thank the other two students, Arthur Batel and Fryderyk Pryjma, as well as the other supervisors, Ted Strandberg and Martin Skoglund, for their support and valuable discussions.

References

[1] 5GAA, "Tele-operated driving - use cases, system architecture and business considerations." 5GAA Automotive Association, 2021.

[2] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[3] Linux Foundation. NETEM. [Online]. Available: https://wiki.linuxfoundation.org/networking/netem

[4] J. den Ouden, V. Ho, T. Van der Smagt, G. Kakes, S. Rommel, I. Passchier, J. Juza, and I. T. Monroy, "Design and evaluation of remote driving architecture on 4G & 5G mobile networks," *Frontiers in Future Transportation*, p. 31, 2022.

[5] R. Liu, D. Kwak, S. Devarakonda, K. Bekris, and L. Iftode, "Investigating remote driving over the LTE network," in *Proceedings of the 9th international conference on automotive user interfaces and interactive vehicular applications*, 2017, pp. 264–269.

[6] M. Hofbauer, C. B. Kuhn, G. Petrovic, and E. Steinbach, "TELE-CARLA: An open source extension of the CARLA simulator for teleoperated driving research using off-the-shelf components," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 335–340.

[7] ISO, "ISO/PAS 21448:2019 - Road vehicles – Safety of the intended functionality," 2019.

[8] F. Warg, M. Skoglund, and M. Sassman, "Human Interaction Safety Analysis Method for Agreements with Connected Automated Vehicles," in *IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, Sep. 2021, pp. 01–07.

[9] G. Jahangirova, A. Stocco, and P. Tonella, "Quality metrics and oracles for autonomous vehicles testing," in *14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2021, pp. 194–204.

[10] SAE, "AVSC best practice for metrics and methods for assessing safety performance of automated driving systems (ads)," 2021.

[11] M. N. Sharath and B. Mehran, "A literature review of performance metrics of automated driving systems for on-road vehicles," *Frontiers in Future Transportation*, p. 28, 2021.

[12] SAE, "SAE J2944 operational definitions of driving performance measures and statistics," 2015.

[13] K. Vogel, "A comparison of headway and time to collision as safety indicators," *Accident analysis & prevention*, vol. 35, no. 3, pp. 427–433, 2003.

[14] G. Breyer, "Safe distance between vehicles," https://www.cedr.eu/docs/view/60794fa6cf0c0-en, p. 9, 2010.

[15] L. Nussbaum and O. Richard, "A comparative study of network link emulators," in *Communications and Networking Simulation Symposium (CNS'09)*, 2009.

[16] G. Stefaniak Niemiec and T. Silva Weber, "Design and implementation of a fault injection prototype on an autonomous vehicle simulator," Universidade Federal Do Rio Grande So Sul, 2021.

[17] S. Malik, M. A. Khan, and H. El-Sayed, "CARLA: Car learning to act—an inside out," *Procedia Computer Science*, vol. 198, pp. 742–749, 2022.

[18] Trafikverket, "The theory test and practical driving test," https://bransch.trafikverket.se/en/startpage/driving-licence/obtaining-a-swedish-driving-licence/the-driving-licence-test-consists-of-two-different-tests/.

[19] G. K. Kountouriotis, P. Spyridakos, O. M. Carsten, and N. Merat, "Identifying cognitive distraction using steering wheel reversal rates," *Accident Analysis & Prevention*, vol. 96, pp. 39–45, 2016.