A Context-aware Failure Detector for VANETs

1st Margarete Sá
Distributed Systems Laboratory (LaSiD)
Postgraduate Program in Mechatronics
Federal University of Bahia, UFBA
Salvador, Brasil
magsa@ufba.br

2nd Sérgio Gorender
Distributed Systems Laboratory (LaSiD)
Postgraduate Program in Mechatronics
Department of Interdisciplinary Computation
Federal University of Bahia, UFBA
Salvador, Brasil
gorender@ufba.br

***Abstract*—Vehicular Ad Hoc Networks (VANETs) are complex and dynamic networks that provide communication between vehicles. Due to its dynamism, it is difficult to adequately manage the resources and services of vehicular networks and to execute ITS (Intelligent Transportation Systems) applications reliably. In particular, applications developed to provide safe driving have strong requirements for vehicular communication, and this communication is provided without guarantees. In this paper, we present a context-aware failure detector service. This service uses Context information and Quality of Context metrics to improve the quality of detections. The concept of context has been developed for pervasive systems. It is used in VANETs to represent the status of a vehicle and its environment, including other vehicles and their communication. The failure detector service has been implemented in a simulation environment based on the Veins framework. The results demonstrate the effectiveness of the failure detector in using context and context quality metrics.**

***Index Terms*—Context-aware, VANETs, Failure Detector.**

I. Introduction

Intelligent Transportation Systems (ITS) applications are being studied to provide safe and comfortable driving and travel on roads and streets. To execute these applications, it is necessary to provide reliable communication between vehicles, especially those developed to provide safe driving. However, providing communication between vehicles is a challenge due to their constant movement, making the network dynamic. To address these challenges, Vehicular Ad-Hoc Networks (VANETS) have been studied, and numerous protocols have been developed [12], [17].

In [10], [13], [14] we presented a context-aware VANET, defining mobility and communicating context. We also presented quality of context metrics, such as Validity Time, Minimal Validity Time, Age, Confidence, and Stability, with their formulas. Context and QoC metrics may be used by network protocols when deciding how to adapt to changes in the environment of vehicles and their communication to provide a better communication service. Context-aware systems have been proposed for pervasive systems, and have been investigated for VANETS in many works [6], [15].

An important service to be included in this context-aware VANET is a failure detector. It aims to monitor vehicles and inform those that are functioning and communicating, as well as those that are suspected of being faulty. The dynamic characteristics of VANETs make it difficult to develop a failure detector in these networks. Despite this, there are some works in this area, such as Cambruzzi et al. ([3]) that present a failure detector for VANETs, using vehicle communication reach.

In this paper, we propose a failure detector protocol based on the FD presented in [3], which runs over our context-aware VANET. Our FD uses context information and QoC metrics to help monitor vehicles. We present our algorithms, discuss their behaviour, and present some simulation results.

The rest of the paper is organized in the following sections: Section II presents related works; Section III presents the system model with the description of the context and some QoC metrics; Section IV presents the failure detector algorithm; Section V shows the prototype implementation and simulation results; and Section VI is the conclusion.

II. Related works

In distributed failure detectors, such as those developed for VANETs, vehicles are monitored through the use of messages. These services may be implemented using the push model, where nodes send heartbeat messages to each other, or the pull model, where monitoring nodes send "*Are-you-alive*" messages to the monitored ones and wait for an "*I-am-alive*" message in return. As there are no guarantees in the delivery of messages, which may be lost or delayed, a failure detector for VANETs is assumed as unreliable, and the failure detector may be wrong when it suspects a fault [4], [7]. These services may also be fully distributed, where nodes monitor all the other nodes, or hierarchical.

Cambruzzi et al. [3] present a failure and connectivity detector that executes among neighbouring nodes in a VANET. Neighbour nodes are near vehicles, considering a limited radius. The protocol sends periodic messages to maintain a neighbourhood list. The failure detector algorithm maintains a list of neighbours suspected of being faulty. A timeout to wait for return messages from the monitored neighbours is calculated, using the distance between the involved nodes.

Abrougui et al. [1] present a fault-tolerant service discovery protocol for vehicular networks, based on a cluster infrastructure. The failure detector exchanges beacon messages in rounds, also determining a time-to-live for each message. The algorithm detects permanent and intermittent failures.

Davoodi et al. [5] present a failure detection methodology for multi-agent systems executing over vehicular networks. The methodology is semi-decentralized and can detect agent faults and those of their nearest neighbours.

Jiaxi Liu et al. [9] present a hierarchical failure detector service for VANETs, where Road Side Units (RSUs) are responsible for monitoring neighbour vehicles. It assumes an unreliable failure detector, which uses mobility information from vehicles in calculating timeouts by the RSU. It is also possible to indirectly detect faulting vehicles, as RSUs send their suspect list to each other.

Byeon et al. ([2]) present a hierarchical failure detector for VANETS where vehicles are monitored by Road Side Units (RSUs) and by other vehicles. It uses estimations of the speed and location of the vehicles, along with a dynamic timeout for receiving heartbeat messages.

In these works, the use of environmental information is ad hoc and not structured. In our work, information is formalized as context information and QoC metrics, which are maintained by a context-aware VANET system. We formalize mobility information as context information and propose a failure detector service that is based on context information and QoC metrics. The failure detector in this paper is distributed and does not use cluster infrastructure.

III. System Model

We assume a vehicular ad hoc network (VANET) composed of autonomous vehicles moving through streets or roads. Vehicles are equipped with sensors, such as GPS and a speedometer, to monitor their position, speed, and direction of movement. The granularity of the information and the periodicity of their monitoring depend on the type of sensors used.

The vehicles communicate with each other through their hardware and software. We assume that there is a connection between two vehicles if they are within the reach of each other's wireless signal. If the vehicles can't communicate directly, the VANET tries to construct a communication route between them. The routing algorithm is out of the scope of this work. Communication occurs by message transfer in both directions. Vehicles maintain their clocks synchronized due to GPS.

Vehicles fail by crashing. In this case, the vehicle stops sending messages. This happens when a vehicle stops functioning or if there is a problem with its communication capabilities. A connection between two vehicles may fail by omission when a message is missing, as the communication is not reliable, and messages can be lost.

A. A Context for VANETs

Context information is maintained in each vehicle by a context monitoring service that executes a continuous monitoring loop. This service monitors information from local sensors, such as GPS and a speedometer, and maintains this information in a local context database. With this information, the system also calculates the direction of vehicle mobility. The monitoring system obtains the same information from remote vehicles using monitoring messages. Together with each local or remote information, the context system maintains timestamps, indicating when the information was obtained [10], [13], [14]. Position and mobility information are used to calculate, for each remote vehicle related to the local one, its relative speed. Based on this information and the timestamps of each information, the system defines the Neighbourhood context.

The Neighbourhood context identifies the vehicles that are neighbours of the local one. Vehicles are assumed as neighbours if their distance is less than the radius of their communication signal [10].

- Neighbour - the remote vehicle is a neighbour;
- Non-Neighbour - the remote vehicle is not a neighbour.

Executing the monitoring system, vehicles also send their Neighbourhood list to each other, together with the timestamp of the last contact they received from these neighbours (Tsp_r^j — timestamp of remote vehicle r in the context).

We also presented the following quality of context metrics:

- Validity Time (VT_{rl}) - this metric represents an estimation of the time during which a remote vehicle (r) may continue to be a neighbour for the local one (l), assuming that the vehicles may maintain the direction and speed of their movement.
- Minimum Validity Time VT_{rl}^{min} – similar to Validity Time, but assuming that vehicles are moving away at maximum speed. It may be considered as a hard Validity Time.
- Timeliness – represents how recent the calculated Validity Time is.
- Age (Age_{rl}) – the age of context information (Age_{rl}^{nei} is the age of the Neighbourhood context relating a remote vehicle r to the local one l) is the difference between the current timestamp and the timestamp when the information has been obtained.
- Trusted Validity Time (TVT_{rl}) – it is the Minimum Validity Time, considering the Age of the metric.

The algorithms of the monitoring system are out of the scope of this work.

IV. Failure Detector algorithm

The context-aware failure detector described in this section is based on the algorithms presented by Cambruzzi et al. [3]. We use the same ideas with some changes, characterizing it as a context-aware failure detector.

The failure detector is distributed in modules, one for each vehicle in the VANET. Performing the failure detector, each vehicle monitors its neighbours, which are identified by the Neighbourhood context, and detects the occurrence of failures. The service is executed in push mode, where each vehicle sends heartbeat messages to its neighbours, which wait for these messages for a period (timeout). If a heartbeat message does not arrive in time, the sender is suspected of failure.

As an unreliable failure detector, the service may suspect vehicles that are executing correctly. We define two types

of fault suspicions. We say that there is a *suspicion* if the service does not receive a heartbeat message from a neighbour before its timeout, and the **Trust Minimal Validity Time** (TVT_{rl}) relating the remote vehicle with the local one is greater than *0*, indicating that the remote vehicle should still be within the reach of their communication radio. In this case, before suspecting the vehicle, the algorithm executes a second monitoring, in the pull mode, sending a *Are-you-alive?* message to the vehicle and waiting for the *I-am-alive* response. The timeout defined for the waiting of the *I-am-alive* message is calculated as $timeout_r = rtt_r + \alpha$, where rtt_r is the round-trip time (period for a message to be transferred from l to r and back), and α is an error value. If TVT_{lr} equals 0, but the **Validity Time** (VT_{rl}) related to the vehicles is still valid, the protocol assumes that r is *weakly suspected*. In this case, it is possible that the vehicle is out of the range of the communication and will soon be assumed as Non-Neighbour.

The Algorithm 1 presents the failure detector protocol. Executing Task T0, the vehicle periodically sends context heartbeat messages to all neighbours. When a vehicle receives a heartbeat message from a remote vehicle r, it executes task T1. In this case, it calculates a new timeout for r, and if it is in the $SuspectList$ or the $WeakSuspectList$, it is removed from it. The timestamp for the received heartbeat message is inserted in the Context.

Task T2 monitors the timeouts for each monitored neighbour vehicle. If a heartbeat message from a remote vehicle r does not arrive during its timeout (β_r), the monitored vehicle is *Suspected* or *WeakSuspected* to be faulty.

Task T3 monitors context information sent by neighbours about their neighbours. It verifies if there is a timestamp for the vehicle r in the context (Tsp_r^j) that is more recent than the timestamp of the last received heartbeat from this vehicle (Tsp_r). It indicates that the heartbeat message from r was lost by the local vehicle but received by some other common neighbour. In this case, if vehicle r is suspected, it is removed from $WeakSuspectedList$ or $SuspectList$.

Task 4 waits for an *I-am-alive* message from a remote vehicle r, in response to an *Are-you-alive* message. It functions as a second monitoring for vehicle r, before suspecting it of being faulty. Task 5: Reply to a received *Are-you-alive* message with an *I-am-alive* message.

In this algorithm, the timeout β_r is calculated in the same way as in [3], as $\beta_r = Q + A_r + \Delta_r$, where Q is the period to send each heartbeat message, A_r is the square mean for the last n delays, and Δ_r is a delay coefficient related to the distance between the local and monitored vehicles, which is indicated in the context.

As an unreliable failure detector, it presents false suspicions when a heartbeat message is lost or late. Due to the dynamic behavior of VANETs, messages are frequently lost or delayed, causing these false suspicions. On the other hand, as the failure detector runs in distributed modules within each car, distinct vehicles may have different views of the monitored vehicles, which can be suspected by some vehicles and not by others. These particular views of the system's state are common

Algorithm 1:

```
1  Task: T0 ;
2  while True do
3      if HeartbeatTime then
4          send Heartbeat(Tsp) to all neighbours
5  Task: T1;
6  while True do
7      if receive Heartbeat() from r then
8          Insert Tsp_r in Context;
9          Calculate β_r (timeout for vehicle r) ;
10         if r ∈ SuspectList then
11             remove r from SuspectList
12         if r ∈ WeakSuspectList then
13             remove r from WeakSuspectList
14 Task: T2 ;
15 while True do
16     if tsp − max(tsp_r, tsp_r^j) > β_r then
17         if isNeighbor(r) and TVT_lr > 0 then
18             send Are-you-alive() to r ;
19             define timeout_r = rtt_r + α ;
20             run T4
21         else
22             if isNeighbor(r) and VT_lr > Tsp then
23                 insert r ∈ WeakSuspectList;
24 Task: T3 ;
25 while True do
26     forall r, isNeighbour(r) ∧ (r ∈
         WeakSuspectlist ∨ r ∈ SuspectList) do
27         if r ∈ SuspectList ∧ TVT_lr > 0
           ∧Tsp_r^j > Tsp_r then
28             remove r from SuspectList ;
29             Calculate β_r (timeout for vehicle r)
30         if r ∈ WeakSuspectList ∧ VT_lr >
           Tsp ∧ Tsp_r^j > Tsp_r then
31             remove r from WeakSuspectList ;
32             Calculate β_r (timeout for vehicle r) ;
33 Task: T4 ;
34 wait (receive I-am-alive() from r) or (tsp > timeout_r)
     if not (receive I-am-alive() from r then
35     insert r in Suspect list;
36 Task: T5 ;
37 if receive Are-you-alive() from r then
38     send I-am-alive(tsp) to r
```

in distributed asynchronous systems because messages can be lost or delivered with varying delays. Using the context, it is possible to differentiate the suspicions, depending on the situation of the vehicles, and in some cases execute a monitoring validation for suspected vehicles before assuming they are indeed suspicious. We utilize quality of context metrics, such as VT_{rl} and TVT_{rl}, intending to qualify the fault suspicions. The context-aware VANET uses the failure detector service to update context information. The service may also be used by ITS applications.

V. The Simulation Environment

For the performance evaluation, we compared our proposed failure detector (FD1) to the failure detector presented in [4] (FD2), in terms of the mean number of false suspicions (FS). We implemented the failure detector algorithms over the Veins framework (Vehicles in Network Simulation) [11], which integrates the network simulator $OMNeT$ [16] and the vehicular mobility simulator $SUMO$ [8]. Our algorithm executes using an implementation of the Context-aware VANET described earlier, with quality of context metrics. To validate the context-aware engineer approach adopted in our failure detector, we disabled the false suspicion confirmation mechanism in this initial performance evaluation.

In this work, we developed simulation scenarios with 4, 10, and 50 vehicles, moving at a maximum speed of $50km/h$, on a straight road and in the same direction. The period to send a heartbeat context message is $0.1s$. Each simulation scenario ran for 300 seconds and was replicated (re-executed) 10 times to obtain the mean values of the number of false suspicions with a 95% confidence level.

TABLE I
False Suspicions - FS

FD	Vehicle Density	FS Mean	FS Confidence Interval
FD1	4	4.03	[2.76, 5.29]
FD2	4	4.05	[2.78, 5.32]
FD1	10	48.79	[46.85, 50.73]
FD2	10	48.68	[46.75, 50.61]
FD1	50	162.54	[158.43, 166.65]
FD2	50	162.66	[158.55, 166.77]

As shown in Table I, the evaluated failure detectors presented similar performance in terms of the number of false suspicions, highlighting the effectiveness of using context and context quality metrics.

VI. Conclusion

We presented a context-aware failure detector that uses the quality of context metrics to qualify fault suspicions. It is an unreliable failure detector, as it may raise false suspicions due to messages being lost or delayed. The algorithm uses the Neighbourhood context to determine which vehicles should be monitored, and QoC metrics, such as Validity Time and Trusted Validity Time, to qualify the fault suspicions. We describe and discuss the algorithm. Some simulation results are used to argue that the context-aware failure detector has similar behavior to existing ones.

The failure detector is an early version that may evolve using other QoC metrics to optimize fault detections, making the detection less unreliable. In future works, we will investigate the use of other QoC metrics with the FD. We will also conduct additional experiments to evaluate our algorithm.

References

[1] Abrougui, K., Boukerche, A., Ramadan, H.: Performance evaluation of an efficient fault tolerant service discovery protocol for vehicular networks. Journal of Network and Computer Applications **35**(5), 1424–1435 (2012)

[2] Byeon, H., Keshta, I., Mahalakshmi, V., Soni, M., Khan, I.R., Maranan, R.: Blockchain based dead reckoning navigation system for fault detection and secure communication over vanet. In: 2023 International Conference on New Frontiers in Communication, Automation, Management and Security (ICCAMS). vol. 1, pp. 1–7 (2023)

[3] Cambruzzi, E., Farines, J.M., Macedo, R., Werner, K.: An adaptive failure detection system for vehicular ad-hoc networks. In: 2010 IEEE Intelligent Vehicles Symposium (IV 2010) (June 2010)

[4] Chandra, T.D., Toueg, S.: Unreliable failure detectors for reliable distributed systems. Journal of the ACM **43**(2), 225–267 (march 1996)

[5] Davoodi, M., Khorasani, K., Talebi, H., Momeni, H.: A robust semi-decentralized fault detection strategy for multi-agent systems: An application to a network of micro-air vehicles. International Journal of Intelligent Unmanned Systems **1**(1), 21–35 (2013)

[6] Dressler, F., Klingler, F., Sommer, C., Cohen, R.: Not all vanet broadcasts are the same: Context-aware class based broadcast. IEEE/ACM Transactions on Networking (2017)

[7] Gorender, S., Macedo, R.J.A.: A dynamically qos adaptable consensus and failure detector. In: Proceedings of The IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2002 - Fast Abstract Track. pp. B80–B81. An extended version has been published in the Proceedings of Brazilian Symposium on Computer Networks and Distributed Systems (SBRC2002), Pages 277–292. (june 2002)

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

[9] Liu, J., Chen, S., Gui, G., Gacanin, H., Sari, H., Adachi, F.: Failure detector based on vehicle movement prediction in vehicular ad-hoc networks. IEEE Transactions on Vehicular Technology **72**(9), 11657–11667 (2023). https://doi.org/10.1109/TVT.2023.3266106

[10] Sá, M., Gorender, S.: Quality of context for vanets: Qoc metrics for connectivity in vanets. In: 2019 18th International Conference on Ad-Hoc Networks and Wireless (ADHOC-NOW). pp. 420–431. Springer (2019)

[11] Sommer, C., German, R., Dressler, F.: Bidirectionally coupled network and road traffic simulation for improved ivc analysis. Mobile Computing, IEEE Transactions on **10**(1), 3–15 (2011)

[12] Souri, A., Zarei, M., Hemmati, A., Gao, M.: A systematic literature review of vehicular connectivity and v2x communications: Technical aspects and new challenges. International Journal of Communication Systems **37** (04 2024). https://doi.org/10.1002/dac.5780

[13] Sá, M., Gorender, S.: Uma análise da confiança na qualidade do contexto em vanets. In: Anais Estendidos do XI Simpósio Brasileiro de Engenharia de Sistemas Computacionais. pp. 56–63. SBC, Porto Alegre, RS, Brasil (2021). https://doi.org/10.5753/sbesc_estendido.2021.18494

[14] Sá, M., Gorender, S.: The age and updating stability for a communication context in vanet. In: 2024 XIV Brazilian Symposium on Computing Systems Engineering (SBESC). pp. 151–156. SBC, Recife, Pe, Brasil (2024). https://doi.org/10.1109/SBESC65055.2024.10771914

[15] Vahdat-Nejad, H., Ramazani, A., Mohammadi, T., Mansoor, W.: A survey on context-aware vehicular network applications. Vehicular Communications **3**, 43–57 (2016)

[16] Varga, A., et al.: The omnet++ discrete event simulation system. In: Proceedings of the European Simulation Multiconference (ESM 2001). pp. 319–324 (2001)

[17] Yogarayan, S., Razak, S.F.A., Azman, A., Abdullah, M.F.A., Ibrahim, S.Z., Raman, K.J.: A review of routing protocols for vehicular ad-hoc networks (vanets). In: 2020 8th International Conference on Information and Communication Technology (ICoICT). pp. 1–7. IEEE (2020)