An improved multiple manoeuver management protocol for platoon mobility in vehicular ad hoc networks

Nigar Anjum Fida1  |  Naveed Ahmad2  |  Yue Cao3,4  |  Mian Ahmad Jan1  |  Gauhar Ali2

1 Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan
2 Department of Information Technology and Computer Science, Pak-Austria Fachhochschule Institute of Applied Sciences and Technology Haripur, Pakistan
3 School of Cyber Science and Engineering, Wuhan University, China
4 Research Institute of Beihang, University in Shenzhen, China

Abstract
Techniques in vehicular ad-hoc networks have been utilised to reduce challenging issues such as pollution, accidents and traffic jams. The platoon management system (PMS) has shown promising results to solve the aforementioned problems. In PMS, vehicles in a platoon-based driving network execute several manoeuvres, e.g. platoon formation, joining, leaving and disruption of platoons. However, existing solutions could not well perform in real time scenario, e.g. handling multiple manoeuvres simultaneously, and fail to provide accurate observation to handle all the possible scenarios of platoon manoeuvres. In this article, an improved multiple manoeuver management protocol (IMMP) is proposed for platoon mobility management. IMMP manages multiple joining and leaving manoeuvres simultaneously, via vehicle-to-infrastructure and vehicle-to-vehicle communications. The performance of IMMP is highlighted by the developed real time simulation tool. Also, the logical flow of IMMP has been presented, by verifying various systems design properties using PROMELA and SPIN validation tool. Simulation results and analysis validate the behaviour of joining and leaving procedures, without infringing the safety of the whole system. This study shows the efficiency of IMMP that works successfully within acceptable manoeuvers duration time.

1 | INTRODUCTION
Vehicular ad-hoc networks (VANETs) [1, 2] have attracted significant attention from industry as well as academia [3]. In VANETs, each participating vehicle behaves as a wireless router or a node to timely exchange information with each other, through wireless technologies [4]. They have been utilised for a wide range of applications, such as human-safety, traffic-flow optimisation and management and infotainment[5].

With VANETs, highway vehicle platooning is an initiative that is a milestone in conveyance industry [6, 7]. A platoon is a group of vehicles that travel in the vicinity of one another, with short inter-vehicular distance. A number of vehicles follow the vehicle in front of group, i.e. a platoon leader, that closely match their speed, brake and manoeuvers. As a result, a highway can arrange space to more vehicles, this increases the road capacity [8, 9]. In general, the most common research issues in vehicle platooning are string stability, communication sustainability [10] and platoon management system (also known as team coordination). Thanks for platoon, the lead vehicle will shoulder the same aerodynamic drag as regular driving, but all following vehicles will be drafting the vehicle in front and therefore experience reduced air resistance. This reduction of aerodynamic drag transforms into greater fuel efficiency and less pollution.

Vehicles in a platoon-based driving network normally execute several manoeuvres that involve the formation, joining, leaving and disruption of platoon. The category manoeuvres describe a series of changes in movements or direction toward an objective. The vehicles in an intra-platoon based driving pattern, perform various manoeuvres within a single platoon. Moreover, it mainly concerns with the multiple joining and leaving of a vehicle in a platoon. However, platoon manoeuvering for multiple vehicle joining and leaving the platoon, has not been adequately investigated in literature. Previous works [11–15] allow only single vehicle and single manoeuver at a time. For instance, a follower vehicle may want to leave the platoon by sending “leave-request” to the platoon leader, whereas the latter may already be engaged in another manoeuver request. In this case, the leader
may send a “leave-reject” message to either dismiss or postpone an incoming leaving manoeuver.

Besides, there are no traces in the literature to formally model and verify multi manoeuvers for safety and security properties. Thus it is important to simulate/emulate platoon management system (PMS) for verifying the safety and duration time. There is also no simulations carried out for single or multi manoeuvers in literature.

In this article, we develop practical platoon management protocol using a decentralised platoon management system. The proposed system moves away from the conventional system of centralised decision towards the platoon leader, which allows only one manoeuver at a time. Instead, the novelty of the proposed platoon management system is to manage multiple manoeuvers at simultaneously. We summarise our contributions as:

- We develop a novel strategy that properly supports multiple “joining” manoeuvers simultaneously on target merge point (TMP) position. The proposed strategy proves that the multiple “joining” manoeuvers are successfully performed without defying the safety of the whole system.
- We design an innovative method that handles multiple “leaving” manoeuvers from an arbitrary locations at a time. Our aim is to satisfy multiple “leaving” requests in case of emergency situation. This is not supported in literature due to centralised decision by the platoon leader (which just supports single request at a time).
- We compute the optimal splits for joining and leaving vehicles, in case, if one vehicle needs to join the platoon while another vehicle needs to leave the platoon at a time. The proposed scheme is able to calculate the desired TMP and the target leave point (TLP), to join and leave the platoon simultaneously. It also ensures that the created gaps refilled again after a successful joining and leaving procedure spontaneously.
- The overall approach is justified by developing a real time simulation tool. Besides, the formal modelling of the platoon management system has been evaluated using SPIN model checker by verifying the formal properties to ensure safety.

The rest of the article is described as follows. Related work is elaborated in Section 2. In Section 3, preliminary of the proposed model has been described. In Section 4, three use cases are discussed with dedicated algorithms. In Section 5, simulation and performance are evaluated. In Section 6, formal modelling and verification are exposed. Finally, Section 7 concludes the article with future directions.

## 2 RELATED WORK

VANETs allow major advances in transportation system [16], e.g. improve travelling comfort, safety and alleviating traffic jam. Literature for platoon manoeuvers management are summarised in Table 1. Further we divide the literature into two sub categories: PMS for personnel vehicles, and commercial vehicles.

### 2.1 PMS for personnel vehicles

In [11], Amoozadeh et al. present a protocol for cooperative adaptive cruise control (CACC) of vehicles. Authors expose only three specific platooning manoeuvers, which are leaving of the leader in platoon, leaving of a follower from middle or end of platoon, and entry at the end of platoon only. Also, this work does not allow more than one manoeuver simultaneously, and does not satisfy real traffic scenarios. Thus during “follower leave manoeuver”, only one vehicle has the permission to leave the platoon at a time.

The works in [12, 17] are based on the multi agent-based system to perform inter-vehicle coordination. To model the automated vehicles for collaborative driving, these works propose a hierarchical driving agent architecture in [18]. To recognise the most favourable collaborative model, four distinct inter-vehicle communication models (hard-centralised, centralised, decentralised and teamwork) are studied. Results show that the teamwork coordination model ensures the flexibility and safety with the least communication cost.
In [13], an application layer protocol is purely proposed for joining manoeuver. It detects numerous instances of interference caused by human-driven vehicles during joining manoeuver.

Bengtsson et al. [19, 20] design advanced interaction protocols to make various platoon manoeuvres. The simulation scenario describes two platoons running on two adjacent lanes concurrently. The roadwork alarming messages will accelerate the platoons to merge into another platoon, with the condition to permit only one vehicle to merge at a time.

2.2 | PMS for commercial vehicles

Nowakowski et al. [14] discuss an accurate functional explanation of CACC operations for heavy vehicles (e.g. trucks). The procedure for platoon split manoeuver (leaving) for vehicles may leave the platoon at any moment from any position, i.e. front, middle or end.

Whereas, the new truck joins the platoon from the end only is considered as the least technically challenging case. However, this precise description of the basic manoeuvres does not provide detailed simulation models.

Based on different distributed coordination schemes among the vehicles in the platoon. Michaud et al. [15] investigated the feasibility of platooning manoeuvering, such as joining and leaving. They also dealt with failure regarding specific perceptual capabilities. A mobile robotic platform is used to imitate platooning circumstances that do not represent to handle real automobile adaptation.

The literature lacks handling multiple manoeuvres simultaneously, suitability for real time applications and extensive demonstration of various use cases of PMS. Therefore, a novel improved platoon management protocol supported by simulation tool is proposed to overcome the existing limitations in handling multiple requests simultaneously. Furthermore, to satisfy the correctness of a system during joining and leaving a platoon safely, we require a formal verification of the system.

2.3 | Formal modelling

Formal modelling [21] is used to make sure that these intelligent controllers in the platoon system never defile safety pre-requisites. We make use of formal modelling methods to ensure system design correctness and high integrity procedures. An essential feature of the formal verification is that the whole platooning system does not violate safety during manoeuver. The works in [22, 23] apply formal verification, to guarantee that the autonomous performance will remain in safe mode. Model checking is a tool that helps to detect the errors in a system model. To provide the automated support for model checking, there have been a number of tools [24, 25].

Simple PROMELA interpreter (SPIN) is a model checking tool to verify leaving and joining manoeuvres of platoon management [26]. It is used for analysing the coordination of concurrent processes, in which a state machine automaton of a system is compared with correctness properties. The design system is modelled in a specification language called protocol/process meta language (PROMELA), to verify the system behaviour [27].

3 | PRELIMINARY

We assume one way and non-interrupted vehicle traffic highway of length “D” with two lanes. One lane is reserved for vehicles platoon, while another is for vehicles not in platoon status, as shown in Figure 1.

Our proposed model uses a decentralised coordination approach. The road side units (RSUs) are deployed along the roadside to collect and deliver a platoon management requests, connect vehicles to the servers on the highway. The RSU is an access points, used together with the vehicles, to allow information dissemination in the roads. These management requests can be captured by nearby RSUs through V2I communication to coordinate the manoeuvre, with joining, or leaving vehicle in a platoon. Besides, vehicles in platooning communicate the control information (known as beacon messages) through V2V communication.

In [28], the study reveals that the V2I is preferred in comparison to the V2V. The reason is its capability to provide information at constant time interval reliably. Also, it can upgrade the delivery rate of management requests, and reduce delivery delay substantially. However, in a decentralised platoon coordination, the platoon leader is still responsible for maintaining the platoon safety. In order to perform manoeuvers request as well as to control platoon driving in case of any emergency, each member vehicle has information about the platoon creation, size and type etc. Here, the platoon formation and communication is beyond the scope of this research, as we are instead interested in platoon manoeuvres management.

In this model, we begin by presenting an autonomous platoon management system. In order to expose the formal modelling to investigate the joining and leaving manoeuvres, each vehicle checks its surroundings and pursues the subsequent instructions from the RSU.
Joining procedure: A vehicle $V$ can join a platoon either at the end or at the centre. $F$ is the follower vehicle that increases the space for the joining vehicle. The RSU $R$ is responsible for managing the join manoeuvre as shown in Figure 2. The joining procedure is as follows:

1) $V$ sends a join request to $R$ that responds with an accept message, which includes the optimal position of $V$ in the platoon.

2) $R$ instructs $F$ to increase a space for letting $V$ in the platoon. Then $F$ decreases its speed, leaves a space for $V$, and informs $R$ about its updated motion.

3) $R$ communicates with $V$ to move in, then $V$ changes lane and notifies $R$. Besides, $R$ informs $F$ to reduce the space and inform $R$ about its updated motion.

4) $R$ communicates the updated information to all the member vehicles.

Leaving procedure: A vehicle $L$ can leave a platoon at any time. The follower vehicle $F$ increases the space, for leaving vehicle to exit from a platoon. The $R$ is responsible for managing the leave manoeuvre as shown in Figure 3. The leaving procedure is as follows:

1) $L$ sends a leave-request to $R$ and waits for the agreement.

2) On receiving the authorisation, $R$ informs $F$ to increase a space for letting $L$ to exit.

3) $F$ slows down and $L$ increases the gap with its preceding vehicle to leave the platoon safely.

4) $L$ changes lane and notifies $R$. Then $R$ instructs $F$ to increase its speed to refill the gap. Later, $F$ notifies $R$ about its updated motion.

5) $R$ communicates the updated information to all the member vehicles.

4 | IMPROVED MULTIPLE MANOEUVRE MANAGEMENT PROTOCOL

The proposed IMMP is composed of three sub algorithms: multiple leaving, multiple joining, multiple leaving and joining respectively. These proposed algorithms cover platoon management system for comprehensive use cases of joining and leaving. Here, the list of notations are summarised in the Table 2.

4.0.1 Multiple leaving manoeuvre

Figure 4 describes the behaviour of a multiple leave manoeuvre. The platoon $P$ consisting of seven member vehicles with platoon leader $X$ depicted in Figure 4(a). Two vehicles $L1$ and $L2$ are the vehicles to leave platoon, and $S1$ and $S2$ are the target split vehicles that are responsible to increase the space for the leaving vehicles to leave the platoon safely. The RSU is responsible for managing the leave manoeuvre procedure. The procedure for a leaving scenario is as follows:

1) $L1$ and $L2$ send a Leave-Rqst message to a nearby installed RSU at a time. Both these vehicles are supposed to be nearer to their destination as shown in Figure 4(a).
TABLE 2  List of nomenclatures

| Notations | Description |
|-----------|-------------|
| $P$       | Platoon name |
| $X_P$     | Platoon leader of platoon P |
| $IX_P$    | Member vehicles identification number |
| $P_D$     | Platoon depth (vehicle position within the platoon) |
| $P_{v-P}$ | Number of vehicles in platoon P |
| $L_{1-REQ}$ | First leave request message of leaving vehicle |
| $L_{2-REQ}$ | Second leave request message of leaving vehicle |
| $L_{-P_v}$ | Leaving vehicle’s position (GPS) |
| $J_{-P_v}$ | Joining vehicle’s position (GPS) |
| $A$       | Sub-platoon name |
| $X_A$     | Platoon leader of sub-platoon A |
| $IX_A$    | Member vehicle identification number of platoon A |
| $B$       | Sub-platoon name |
| $Y_B$     | Platoon leader of sub-platoon B |
| $IX_B$    | Member vehicle identification number of platoon B |
| $C$       | Sub-platoon name |
| $Z_C$     | Platoon leader of sub-platoon A |
| $IX_C$    | Member vehicle identification number of platoon A |
| $P_D(L_{-1-REQ})$ | Platoon depth of leave request message number 1 |
| $P_D(L_{-2-REQ})$ | Platoon depth of leave request message number 2 |
| Split−REQ | Split request message |
| Split−Acept | Split accept message |
| Merge−REQ | Merge request message |
| DS        | Data stream array |
| $n$       | Even numbers |
| $n+1$     | Odd numbers |
| TMP       | Target merging point |

FIGURE 5  Permitting two vehicles joining simultaneously

2) RSU computes the optimal splits, based on the DS array, according to the required situation.
3) The first split occurs after $L_1$ vehicle and the second split occurs after $L_2$ vehicle, as shown in Figure 4(b).
4) After computing the optimal splits, the RSU sends the split request message to $S_1$ at $P_D = 3$, and $S_2$ at $P_D = 7$.
5) On accepting the split request messages, $S_1$ and $S_2$ reduce their speeds and increase the gap from their preceding vehicles, and split $P$ into three sub-platoons $A$, $B$ and $C$, as shown in Figure 4(c).
6) Moreover, RSU sends a unicast message to $S_1$ and $S_2$ as a newly temporary-based platoon leader.
7) The $S_1$ and $S_2$ then send a multicast message to its member vehicles to announce the current temporary platoon leader (named as $Y$ and $Z$ respectively), as shown in Figure 4(c).
8) In this specific case, the platoon $B$ consists of four member vehicles having the platoon leader $Y$, and platoon $C$ consists of one vehicle, which is considered as a member vehicle as well as a leader $Z$. Therefore, it is assumed as a free agent.
9) The RSU sends unicast message to leaving vehicles $L_1$ and $L_2$, to leave from the end of their respective sub-platoons $B$ and $C$.
10) After a successful safe leaving, the RSU is updated accordingly, as illustrated in Figure 4(c).
11) Finally, RSU sends merge-request messages to all the three platoon leaders to merge again, as shown in Figure 4(d). The three sub-platoons merge into one platoon as platoon $P$ emerges with platoon leader $X$, as shown in Figure 4(e).

4.0.2  Multiple joining manoeuver

The platoon $P$ consisting of five member vehicles having a platoon leader $X$, as depicted in Figure 5(a). In the scenario, two vehicles $J_1$ and $J_2$ are the joining vehicles, and $S_1$ and $S_2$ are
the target split vehicles to increase the space for the vehicles to join the platoon safely. The RSU is responsible for managing the joining manoeuver procedure as follows:

1) \( J_1 \) and \( J_2 \) send a Join-Rqst message to a nearby installed RSU at a time for joining the platoon as shown in Figure 5(a).
2) The RSU computes the optimal splits according to the platoon range.
3) The first split occurs from lines 6 to 10 in Algorithm 2. If the platoon range is even, then the split is \( \frac{s}{2} \). In case of odd range, the split is \( \frac{s+1}{2} \), considering the floor value. The RSU then performs the split by sending a split request message to \( J_1 \) at \( R_3 = 3 \), as shown in Figure 5(b).

---

**Algorithm 1** Multiple leaving manoeuver

1: \( P = [r_1 ... r_s] \)
2: \( X_0 = r_1 \)
3: \( ID_{s_0} \) with \( s \in [1 ... s] \)
4: At \( t_1, L_1\_REQ \) and \( L_2\_REQ \rightarrow RSU \)
5: RSU compute: Dual optimal split
6: Set first optimal split = after vehicle \( L_1\_REQ \)
7: Set second optimal split = after vehicle \( L_2\_REQ \)
8: Send Split–REQ to vehicle: \( P (L_1\_REQ) + 1 \)
9: Send Split–REQ to vehicle: \( P (L_2\_REQ) + 1 \)
10: Send Split–Acpt \( \rightarrow RSU \)
11: Set: \( \{A, B, C\} \) \( \leftarrow P \)
12: \( A = [r_1 ... r_s] \)
13: \( X_A = r_1 \)
14: \( ID_{va} \) with \( a \in [1 ... l] \)
15: \( B = [r_{a+1} ... r_s] \)
16: \( Y_B = r_{a+1} \)
17: \( ID_{vb} \) with \( b \in [l + 1 ... m] \)
18: \( C = [r_{m+1} ... r_s] \)
19: \( Z_c = r_{m+1} \)
20: \( ID_{va} \) with \( c \in [m + 1 ... l] \)
21: \( ID_{va}(L_2\_REQ) \in A \) and \( ID_{va}(L_2\_REQ) \notin B \)
22: RSU send unicast message to: \( ID_{va}(L_2\_REQ) \)
23: \( \triangleright \) to leave as a tail from \( X_A \)
24: RSU: send unicast message to: \( ID_{va}(L_2\_REQ) \)
25: \( \triangleright \) to leave as a tail from \( Y_B \)
26: Update DS:
27: RSU: send Merge–REQ to: \( X_A, Y_B \) and \( Z_c \)
28: \( \triangleright \) to merge again
29: if \( X_A, Y_B \) and \( Z_c \) then
30: Set: \( P \) \( \leftarrow \{A, B, C\} \)
31: else
32: SET: \( \{A, B, C\} \) \( \leftarrow P \)
33: end if

---

**Algorithm 2** Multiple join manoeuver

1: \( P = [r_1 ... r_s] \)
2: \( X_0 = r_1 \)
3: \( ID_{vp} \) with \( p \in [1 ... s] \)
4: At \( t_1, J_1\_REQ \) and \( J_2\_REQ \rightarrow RSU \)
5: RSU compute: Dual optimal split
6: First Split:
7: if \( (P_c \_P = s) \) then
8: Set optimal split = \( \frac{s}{2} \)
9: if \( (P_c \_P = s + 1) \) then
10: Set optimal split = \( \frac{s+1}{2} \)
11: end if
12: Send split–REQ to vehicle: \( P (Y) + 1 \)
13: Second split:
14: if \( (P_c \_P = s) \) then
15: Set optimal split = \( \frac{s}{2} \)
16: if \( (P_c \_P = s + 1) \) then
17: Set optimal split = \( \frac{s+1}{2} \)
18: end if
19: send Split–Acpt \( \rightarrow RSU \)
20: Set: \( \{A, B, C\} \) \( \leftarrow P \)
21: \( A = [r_1 ... r_s] \)
22: \( X_A = r_1 \)
23: \( ID_{va} \) with \( a \in [1 ... l] \)
24: \( B = [r_{a+1} ... r_s] \)
25: \( Y_B = r_{a+1} \)
26: \( ID_{vb} \) with \( b \in [l + 1 ... m] \)
27: \( C = [r_{m+1} ... r_s] \)
28: \( Z_c = r_{m+1} \)
29: \( ID_{va} \) with \( c \in [m + 1 ... l] \)
30: RSU send unicast message to: \( ID_{va}(J_1\_REQ) \)
31: \( \triangleright \) to leave as a tail from \( X_A \)
32: RSU send unicast message to: \( ID_{va}(J_2\_REQ) \)
33: \( \triangleright \) to leave as a tail from \( Y_B \)
34: Update DS:
35: RSU send Merge–REQ to: \( X_A, Y_B \) and \( Z_c \)
36: \( \triangleright \) to merge again
37: if \( X_A, Y_B \) and \( Z_c \) then
38: Set: \( P \) \( \leftarrow \{A, B, C\} \)
39: else
40: SET: \( \{A, B, C\} \) \( \leftarrow P \)
41: end if
42: end if
43: end if
Algorithm 3 Multiple joining and leaving manoeuvre

1: \( P = [r_1 \ldots r_s] \)
2: \( X_A = r_1 \)
3: \( \text{ID}_s, \text{with p} \in [1 \ldots s] \)
4: At \( h_i \), \( f_{req} \text{ and } l_{req} \text{ to RSU} \)
5: RSU computes: Dual optimal split
6: First split:
7: \( \text{if} \ (l_{-P_1} > f_{-P_1}) \text{ and } (l_{-P_1} \geq n - 1) \text{ then} \)
8: \( \text{Set optimal split} = \text{after vehicle } l_{-P_1} \text{ to } \text{RSU} \)
9: \( \text{end if} \)
10: go to step 14
11: \( \text{if} \ (l_{-P_1} < f_{-P_1}) \text{ and } (l_{-P_1} \geq i + 2) \text{ then} \)
12: \( \text{Set optimal split} = \text{after vehicle } l_{-P_1} \text{ to } \text{RSU} \)
13: \( \text{end if} \)
14: RSU:Send First Split-REQ to vehicle: \( \text{ID}_{P+1} + 1 \)
15: Second split:
16: \( \text{if} \ (P_{-P} = s) \text{ then} \)
17: \( \text{Set optimal split} = \frac{1}{2} X \)
18: \( \text{if} \ (P_{-P} = s + 1) \text{ then} \)
19: \( \text{Set optimal split} = \frac{s+1}{2} [X] \leftrightarrow n \geq a < s + 1 \text{ then} \)
20: \( \text{end if} \)
21: Send Split-REQ to vehicle: \( \text{ID}_{P+1} + 1 \)
22: send Split-Accept to RSU
23: Set: \( \{A, B, C\} \leftarrow P \)
24: \( A = [r_1 \ldots r_s] \)
25: \( X_A = r_1 \)
26: \( \text{ID}_s, \text{with p} \in [1 \ldots s] \)
27: \( B = [r_{s+1} \ldots r_a] \)
28: \( Y_B = r_{s+1} \)
29: \( \text{ID}_b, b \in [r_{a+1} \ldots s] \)
30: \( C = [r_{s+1} \ldots r_s] \)
31: \( Z_A = r_{a+1} \)
32: \( \text{ID}_s, \text{with c} \in [a + 1 \ldots s] \)
33: RSU sends unicast message to: \( \text{ID}_s, (l_{-P_1}) \)
34: \( \text{to leave as a tail from } X_A \)
35: RSU sends unicast message to: \( \text{ID}_s, (l_{-P_1}) \)
36: \( \text{to leave as a tail from } Y_B \)
37: Update DS:
38: RSU sends Merge-REQ to: \( X_B, Y_B \text{ and } Z_B \)
39: \( \text{to merge again} \)
40: \( \text{if } X_A, Y_B \text{ and } Z_B \text{ then} \)
41: \( \text{Set: } P = \{A, B, C\} \)
42: \( \text{else} \)
43: \( \text{SET: } \{A, B, C\} \leftarrow P \)
44: \( \text{end if} \)
45: \( \text{end if} \)

4.0.3 Multiple joining and leaving manoeuvre

Figure 6 represents case-1 and Figure 7 depicts the case-2. The platoon \( P \) consisting of seven member vehicles having a platoon leader \( X \), is depicted in Figure 6(a) and Figure 7(a). In this case, one vehicle \( J \) needs to join the platoon while another vehicle \( L \) needs to exit from the platoon, simultaneously. \( S1 \) and \( S2 \) are the target split vehicles that will increase the space, for the joining vehicle \( J \) to join the platoon and leaving vehicle \( L \) to leave the platoon safely. The RSU is responsible for managing the joining manoeuvre procedure as follows:

1) \( J \) and \( L \) notify the nearby installed RSU by sending a Join-Rqst message and Leave-Rqst message, as shown in Figure 6(a).
2) The RSU computes the optimal splits according to the location of leaving and joining vehicles through GPS.
3) The first optimal split of the platoon \( P \) has two conditions:
   \( (l_{-P_1} > f_{-P_1}) \text{ and } (l_{-P_1} \geq n - 1) \) explains that position of leaving vehicle must be higher than that of the joining.
vehicle position. The position of leaving vehicle must be greater or equal to \( n - 1 \). Here, \( n \) is the total number of vehicles in a platoon, shown in Figure 6(b).

- \((L_{<} P_{i} < f_{<} P_{j}) \land (L_{>} P_{i} \geq i + 2)\) explains that the position of joining vehicle must be higher than that of the leaving vehicle. The position of leaving vehicle must be greater or equal to \( i + 2 \). For instance, \( i \) is the first vehicle in a platoon, as shown in Figure 7(b).

4) The RSU then performs the first split by sending a Split-Request message to \( S_1 \) at \( P_{D} = 4 \), as shown in Figure 6(b), and \( S_2 \) at \( P_{D} = 7 \), as shown in Figure 7(b).

5) The second split occurs according to the platoon range. If the platoon range is even, then the split is \( \frac{n}{2} \). In case of odd range, the split is \( \frac{n+1}{2} \), considering the floor value, as shown in Figures 6(b) and 7(b).

6) The RSU, then performs the second split by sending a Split-Request message to \( S_2 \) at \( P_{D} = 4 \), as shown in Figure 6(b), and \( S_1 \) at \( P_{D} = 7 \), as shown in Figure 7(b).

7) On accepting the Split-Rqst messages, \( S_1 \) and \( S_2 \) reduce their speeds and increase the gap from their preceding vehicles, then split \( P \) into three sub-platoon \( A \), \( B \) and \( C \), as shown in Figures 6(c) and 7(c).

8) Moreover, RSU sends unicast message to \( S_1 \) and \( S_2 \) as newly temporary-based platoon leader.

9) The \( S_1 \) and \( S_2 \) then send multicast message to their member vehicles, to announce the current temporary platoon leader (named as \( Y \) and \( Z \)), shown in Figures 6(c) and 7(c).

10) The RSU permits \( L \) to leave from the end in sub-platoon \( A \). Similarly, it permits \( J \) to join at the target merge point (TMP) in the sub-platoon \( B \).

11) After a successful safe joining and leaving, the RSU is updated accordingly with the condition change.

12) The RSU sends a merge-request message to all the three platoon leaders to merge again as shown in Figures 6(d) and 7(d).

13) After merging, the platoon \( P \) emerges with a platoon leader \( X \), as shown in Figures 6(e) and 7(e).

4.1 Discussion

The main motivation of our design is to reduce the signalling overhead, rather than processing delay for each of vehicle to join/leave platoon. If the number of vehicles in group is growing, the overhead may pose challenge on synchronisation in time domain, and the communication cost. In practical, the communication delay is normally less than 1s for small group of vehicle upon each joining/leaving action via 5G. Given that there are large number of vehicles, the per-request processing nature will in deed create engineering aspect problem to maintain the safety of platoon. Besides, the vehicle group movement can be easily estimated by hacking any one of them. Therefore, reducing the controlling signalling exchange frequency may also reduce the possibility that the vehicle group movement can be easily estimated.

5 Simulation and Results

We have developed a simulation tool that is available on github[29] for the community for further improvements.

5.1 Simulation setup

We have built a simulation setup, consists of a straight two-lane unidirectional traffic highway with the speed limit of 70 km/h, as illustrated in Figure 8. We suppose that vehicles are identical passenger cars and are platooned-enabled. The desired range of the platoon is up to 7 vehicles. The leftmost lane (lane 1) is restricted to platoon cars, and rightmost lane (lane 2) is used for non-platoon cars. Cars can perform different
platooning manoeuvres that are usually dictated by the RSUs servers around the highway. For instance, platoon cars running on dedicated lane 1 as long as they are member of a platoon. Upon leaving, they move to lane 2. Similarly, each car on lane 2 moves to lane 1 at some target point, by initiating a joining manoeuvre.

We suppose that all platooning cars can compute their own locations periodically, using an on-board unit (OBU) and a wireless communication (IEEE 802.11p) device. When an event occurs, beacon messages and manoeuvres request messages are disseminated in platoon accordingly. The V2V uses geocast messages, i.e. cooperative awareness message (CAM), and V2I uses unicast messages, i.e. decentralised environmental notification message (DENM). The RSU collects all the beacon messages from the vehicles, and stores it in memory as a data stream (DS) array structure in Table 3. Moreover, both the RSU and platoon leader stores, manages and updates the platoon configuration.

5.2 | Multiple leave manoeuvre

The simulation shows various parameters such as manoeuver running time, platooning running time and also highlights the RSU connectivity results. The Figure 12 illustrates the more realistic implementation of multiple leave manoeuvre. In Figure 12(a), a platoon of 7 vehicles running on lane 1. Two vehicles with ID 1 and ID 5 need to exit from a platoon and sends a leave request to the RSU. In Figure 12(b), after connecting with the RSU, the RSU starts processing and compute the optimal split to exit from the platoon safely. Then in the Figure 12(c), the multiple leaving manoeuvres have been successfully performed. Finally, The RSU ensures that the spacing decreases after a leaving procedure.

5.3 | Multiple joining manoeuvre

The simulation results of multiple joining manoeuvre are show in Figure 13. In Figure 13(a), a platoon of five vehicles running on lane 1. Two vehicles with ID 5 and ID 6 running on lane 2 want to join a platoon and sends a join request to the RSU. In Figure 13(b), after a vehicle connects with the RSU, the latter starts processing and compute the optimal split to join the platoon safely. Then the Figure 13(c) shows the successful joining of platoon. Finally, the RSU ensures that the gap decreases after a joining process.

5.4 | Multiple join and leave manoeuvre

The multiple join and leave manoeuvres are shown in Figure 14. In Figure 14(a), a platoon of 7 vehicles running on lane 1. One vehicle with ID 2 exits from a platoon by sending a leave request to the RSU. Similarly another vehicle ID 7 running on lane 2 wants to join a platoon and sends a join request to the RSU. After a vehicle connects with the RSU, the latter starts processing and compute the optimal split to join and leave the platoon safely as shown in Figure 14(b). Then the Figure 14(c), shows the successful leaving and joining with a platoon. Finally, the RSU ensures that the created gaps refilled again after a joining and leaving procedure.

5.5 | Duration of multiple manoeuvres

This sections provides the performance of the platoon during the application runtime by emphasizing on the execution time of manoeuvres. The Figure 9, depicts the average duration time of multiple manoeuvres in a 7-vehicle platoon running with speed of 70 km/h. We differentiate four types of multiple manoeuvres: joining, leaving, join and leave (case1) and join and leave (case2). The simulation results show that multiple joining takes 5 seconds whereas multiple leaving takes 7 seconds. On the hand, both the cases of multiple join and leave takes almost the same duration time, i.e. 10 s.
5.6 | Comparison

This section compares the simulation results of the proposed work with literature as depicted in Figure 10. The average duration time by our proposed protocol (IMMP) is much lower in comparison to Bruno Ribeiro [30], M. Segata [13] and Amoozadeh [11]. In the joining manoeuvre as shown in Figure 10(a), IMMP has a less duration time and hence outperforms the existing ones. Whereas, in leaving manoeuvre shown in Figure 10(b), the IMMP has a better performance comparative to [30], though IMMP and [11] have equal duration time but still IMMP is better than [11] because it takes 7 s for a single manoeuvre whereas in our case dual manoeuvre takes 7 s. Moreover, in join and leave (case1 and case2) manoeuvres there is no protocol in literature to compare with as we are the first to do it and is the benchmark for the research community as depicted in Figure 10(c,d).

6 | FORMAL MODELLING, VERIFICATION AND ANALYSIS

The proposed work formally modelled using PROMELA language and SPIN formal modelling tool. The code is available on the github for the community for further improvements.

6.0.1 | Joining manoeuvre

The Figure 11 describes the procedure of joining manoeuvre. It depicts the coordinated actions performed by the leader (RSU), and the joining and the follower vehicles. In PROMELA, vehicles are considered as an independent processes.

In our proposed model, The following declares the joining vehicle, RSU and follower vehicle processes in Promela language as:

```
active proctype joining_vehicle()
active proctype RSU()
active proctype follower()
```

The keyword active is a prefix for proctype declarations. In other words, it defines a set of processes that are required to be active in the initial state. The proctype declares the new process behaviour.

The joining process that consists of an idle state, request state, wait state and joining state. The Buchi automaton [31] of a joining process is shown in Figure 11(a). Initially, the vehicle is an idle state. When the vehicle request for joining the platoon then the joining vehicle makes a transition to request state from idle state. In the request state, if the permission is granted, the vehicle transits to the wait state. In the wait state, the vehicle waits
FIGURE 11 State machines for joining manoeuver

for the required space in the platoon to join in. Once the space has been created, it makes transition to joining state from wait state. Finally, when the joining process is successfully satisfied, then it moves back to idle state.

The RSU process consists of an idle state, compute-position state, wait-gap state and End state. The Buchi automaton of leader process is shown in Figure 11(b). In the compute-position state, the RSU checks the optimal joining position where the joining vehicle has to join in. If the compute-position is done, it makes a transition from compute-position state into wait-gap state. In the wait-gap state, the RSU has to wait for the gap which is created by the follower for the joining vehicle. The RSU process makes a transition to End state if it is notified by the follower that the gap has been created.

The follower process consists of idle state, creating-gap state and End state. The Buchi automaton of a follower process is shown in Figure 11(c). The initial state is an idle state. When the follower vehicle receives a create-gap request from the leader then there is transition from idle state to creating-gap state. Once the required space has been created by the follower vehicle, then it moves to End state.

Furthermore, the proposed model consists of channel process, i.e. the transfer of messages between process is known as message channels. We have defined four channels. Channels are declared using the keyword chan. In the joining manoeuver model, the following is the declaration of four channels that can pass messages with multiple fields:

- `chan join = [2] of {mtype, pid};`
- `chan p = [2] of {mtype, pid};`
- `chan q = [2] of {mtype, pid};`
- `chan gap = [2] of {mtype, pid};`

These channels can store up to two messages, each consisting of two fields of the types listed as mtype and pid. A mtype declaration allows for the introduction of symbolic names for constant values. There can be multiple mtype declarations in a verification model. The declaration of mtype variable with four types of messages given below:

- `mtype = created, done, joining, space;`

A pid is a predefined, local, read-only variable that stores the instantiate number of the executing process.

6.0.2 Leaving manoeuver

The Figure 15 describes the procedure of leaving manoeuver. It also depicts the coordinated actions performed by the RSU, leaving and the follower vehicles. In our proposed model, the leaving vehicle, the leader (RSU) and the follower processes are declared in PROMELA modelling language as:

- `active proctype leaving_vehicle();`
- `active proctype RSU();`
- `active proctype follower();`

The vehicles are considered as an independent processes. These processes are describe as follows:

In our proposed model, there is one leaving process and consists of an idle state, request state, wait state and leaving state. The Buchi automaton of a leaving process is shown in Figure 15(a). Initially, the vehicle is an idle state. When the vehicle request for leaving the platoon then the leaving vehicle makes a transition to request state from idle state. In the request state, if the permission is granted, the vehicle transits to the wait state. In the wait state, the vehicle waits for the required space in the platoon to exit. Once space has been created, it makes the transition to leaving state from wait state. Finally, when the leaving process is successfully satisfied, then the vehicle moves back to the idle state.

The RSU process consists of an idle state, compute-position state, create-wait-gap state, done state and end-wait-gap state. The Buchi automaton of RSU process is shown in Figure 15(b). In the compute-position state, the RSU checks the optimal leaving position where the leaving vehicle has to exit. When the compute-position is done, it makes a transition from compute-position state to create-wait-gap state. In the create-wait-gap state, the RSU has to wait for the gap which is created by the follower for the leaving vehicle to exit. Then the RSU process makes a transition to done state if it is notified by the follower that the gap has been created. When the leaving vehicle successfully exits from the platoon then it makes the transition to End-wait-gap state, i.e. the RSU has to wait for the follower to close the gap.

The follower process consists of idle state, create-gap state, end state and end-closed-gap state. The buchi automaton of a
FIGURE 12  Simulation snapshots of multiple leaving

FIGURE 13  Simulation snapshots of multiple joining

FIGURE 14  Simulation snapshots of multiple joining and leaving
follower process is shown in Figure 15(c). The initial state is an idle state. When the follower vehicle receives a create-gap request from the RSU, then there is a transition from idle state to creating-gap state. Once the required space has been created by the follower vehicle to let the leaving vehicle exit from the platoon, then it moves to done state. The platoon returns to a normal state by decreasing the space after a leaving process. Therefore the transition moves to End-closed-gap state from done state, and finally, the maneuver ends, and the transition goes back to idle state.

Moreover, In the leaving maneuver model we have defined six channels. The following are the declaration of a six channels that can pass messages with multiple fields:

```
chan join = [2] of { mtype, pid};
chan p = [2] of { mtype, pid};
chan q = [2] of { mtype, pid};
chan gap = [2] of {mtype, pid};
chan r = [2] of { mtype, pid};
chan k = [2] of { mtype, pid};
```

These channels can store up to two messages, each consisting of two fields of the types listed as mtype and pid. The declaration of mtype variable with six types of messages given below:

```
mtype = (created, done, leaving, space, leave, closed_space);
```

Whereas, a pid variable is used to store the instantiate number of the running process.

6.1 | Formal modelling

Now we can carry out simulation of joining and leaving using SPIN. The Figure 16 shows joining simulation result. We have run one joining, one RSU and one follower process concurrently. Similarly, Figure 17 shows leaving simulation. In which we have run one leaving, one RSU and one follower processes concurrently. In both results, the vertical line represents the execution path of the three processes. The boxes on the vertical line show the execution step of the process. The cross arrow between boxes shows message passing between process.

6.2 | Verification using SPIN

We herein present the formal verification results of the model properties made with the model checking. It is used to verify the safety and liveness properties of a system, and it generated
from a linear temporal logic formula (LTL). The LTL formula is used to specify properties that must be proved by the model. We have checked the following LTL formulas against our model, to determine the absence of acceptance cycles to verify the correctness of the system.

1) Joining

   Property C1: \( \langle\rangle (\text{joining}_{-}\text{vehicle} @ \text{joining}_{-}\text{state}) \)

   The SPIN structure of property C1 automaton is shown in Figure 18. This property states that there exists a joining state for joining vehicle. The vehicle will join the platoon when it requests to join. It states that whenever the joining-request is satisfied, then the joining vehicle go to joining-state.

   Property C2: \( \langle\rangle (\text{joining}_{-}\text{vehicle} @ \text{request}_{-}\text{state}) \rightarrow \langle\rangle (\text{RSU} @ \text{compute}_{-}\text{position}) \)

   The automaton structure of property C2 in SPIN model checker is shown in Figure 19. This property verifies that when a joining vehicle enters into the request-state, then eventually the RSU makes a transition to compute-position state. It states that whenever the joining-request is satisfied, eventually the compute-position state will be satisfied.

   Property C3: \( \langle\rangle (\text{RSU} @ \text{wait}_{-}\text{gap}) \rightarrow \langle\rangle (\text{follower} @ \text{creating}_{-}\text{gap}) \)

   The structure of property C3 automaton in SPIN is shown in Figure 20. This property states that when the leader is in the wait-gap state, then finally the follower will start increasing the space to its front vehicle to create a gap. We verify this property to show that whenever the leader is in the wait-gap state, i.e. has computed the joining location. Then eventually the follower will receive a creating gap command and will increase space for the joining vehicle.

   Property C4: \( \langle\rangle (\text{RSU} @ \text{Done}) \)

   The SPIN structure of property C4 automaton is shown in Figure 21. It states that eventually, the RSU will go to done state. That is, RSU will allow the joining vehicle to join the platoon. In simple words, whenever the follower successfully created the gap to its front vehicle. Then eventually the RSU will not remain in the wait-gap state. It makes the transition from the wait-gap state into done state and grant the permission to joining vehicle to join in.

2) Leaving

   Property C1: \( \langle\rangle (\text{leaving}_{-}\text{vehicle} @ \text{leaving}_{-}\text{state}) \)

   The SPIN automaton of property C1 is shown in Figure 22. This property states that there exists a leaving state for leaving vehicle. The vehicle will leave the platoon when it requests to join. It states that whenever the leaving-request is satisfied, then the leaving vehicle go to leaving-state.

   Property C2: \( \langle\rangle (\text{leaving}_{-}\text{vehicle} @ \text{request}_{-}\text{state}) \rightarrow \langle\rangle (\text{RSU} @ \text{compute}_{-}\text{position}) \)

   The overall structure of property C2 automaton in SPIN is depicted in Figure 23. This property verifies that when a leaving vehicle enters into the request-state, eventually the leader vehicle will make a transition to compute-position state. It states that whenever the leaving-request is satisfied, eventually the compute-position state will be satisfied.
Property C3: \[\{!(\text{leaving\_vehicle} @ \text{leaving\_state}) \rightarrow <> \text{(follower} @ \text{closed\_gap})\}\]

The property C3 automaton in SPIN is shown in Figure 24. It verifies that when the leaving vehicle enters into the leaving-state, then eventually the follower vehicle makes a transition to closed-gap state and will perform the action. It states that whenever the leaving-state is satisfied, eventually the follower closed-gap-request will be satisfied.

Property C4: \[\{!(\text{follower} @ \text{Done}) \rightarrow <> \text{(RSU} @ \text{closed\_gap})\}\]

In SPIN, the automaton of property C4 is shown in Figure 25. We verify this property to show that whenever the follower is in done-state (e.g. it has successfully increased the space for leaving vehicle to exit), eventually, the RSU will issue the closed gap command and will act accordingly.

7 | CONCLUSION AND FUTURE WORKS

This article highlighted the importance of PMS for VANETs, and proposed an IMMP. The IMMP applies a decentralised approach to solve the existing research problem in PMS e.g., by allowing and handling multiple manoeuvres simultaneously. The modelling and verification of IMMP has been carried out using the design language PROMELA and the validation tool SPIN. Simulation results have clearly shown that the proposed decentralised manoeuvre protocol works efficiently within an acceptable manoeuvres duration time. The formal verification has also clearly demonstrated the safety and security properties of IMMP.

It is submitted that in single manoeuvre it is obvious that a leave reject situation occur. This is the motivation for us to work on multiple manoeuvres. However, the focus of this publication to design leave and joining algorithms/protocols for multiple manoeuvres. In the future, it can be explored to monitor the leave-reject and joining-reject use cases.

ACKNOWLEDGEMENT

This work is supported by Guangdong Basic and Applied Basic Research Foundation (2019A1515110238).

REFERENCES

1. Rasheed, A., et al.: Vehicular ad hoc network (vanet): A survey, challenges, and applications. In: Vehicular Ad-Hoc Networks for Smart Cities, pp. 39–51. Springer, Berlin, Heidelberg (2017)
2. Sultan, S.A.I., et al.: A comprehensive survey on vehicular ad hoc network. J. of Net. and Comput. Appl. 37, 380–392 (2014)
3. Liang, W., et al.: Vehicular ad hoc networks: architectures, research issues, methodologies, challenges, and trends. Int. J. of Distr. Sensor Net. 11(8), 745303 (2015)
4. Fida, N., et al.: Performance analysis of vehicular adhoc network using different highway traffic scenarios in cloud computing. In: International Conference on future intelligent vehicular technologies, pp. 157–166. Springer, Berlin, Heidelberg (2016)
5. Cunha, F., et al.: Data communication in VANETs: Protocols, applications and challenges. Ad Hoc Net. 44, 90–103 (2016)
6. Jia, D., et al.: A survey on platoon-based vehicular cyber-physical systems. IEEE Commun. Surveys Tut. 18(1), 263–284 (2016)
7. Bergenhem, C., et al.: Overview of platooning systems. Presented at Proceedings of the 19th ITS World Congress, Vienna, 22–26 October 2012
8. Axelsson, J.: Safety in vehicle platooning: A systematic literature review. IEEE Trans. Intell. Transport. Syst. 18(5), 1033–1045 (2017)
9. Kulla, E., et al.: A survey on platooning techniques in VANETs. In: Conference on Complex, Intelligent, and Software Intensive Systems, pp. 650–659. Springer, Berlin, Heidelberg (2018)

10. Latif, S., et al.: A comparative study of scenario-driven multi-hop broadcast protocols for VANETs. Veh. Commun. 12, 88–109 (2018)

11. Amoozadeh, M., et al.: Platooning with cooperative adaptive cruise control enabled by VANET. Veh. Commun. 2(2), 110–123 (2015)

12. Halé, S., Laumonier, J., Chaib-Draa, B.: A decentralized approach to collaborative driving coordination. In: Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems, pp. 453–458. IEEE, Piscataway, NJ (2004)

13. Segata, M., et al.: Supporting platooning maneuvers through ivc: An initial protocol analysis for the join maneuver. In: 2014 11th Annual Conference on Wireless On-demand Network Systems and Services, pp. 130–137. IEEE, Piscataway, NJ (2014)

14. Nowakowski, C., et al.: Operational concepts for truck maneuvers with cooperative adaptive cruise control. Trans. Res. Rec.: J. of the Trans. Res. Board 2559, 57–64 (2016)

15. Michaud, F., et al.: Coordinated maneuvering of automated vehicles in platoons. IEEE Trans. on Intel. Trans. Syst. 7(4), 437–447 (2006)

16. Arshad, M., et al.: A survey of local/cooperative-based malicious information detection techniques in VANETs. EURASIP J. on Wirel. Commun. Net. 2018(1), 62 (2018)

17. Halé, S., Chaib-draa, B.: A collaborative driving system based on multi-agent modelling and simulations. Trans. Res. Part C: Emerg. Technol. 13(4), 320–345 (2005)

18. Halé, S., et al.: Architectures for Collaborative Driving Vehicles: From a Review to a Proposal. Université Laval, Quebec (2003)

19. Bengtsson, H.H., et al.: Interaction protocol for highway platoon merge. In: 2015 IEEE 18th International Conference on Intelligent Transportation Systems (ITSC), pp. 1971–1976. IEEE, Piscataway, NJ (2015)

20. Englund, C., et al.: The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles. IEEE Wirel. Commun. 23(4), 146–152 (2016)

21. Lucke, M., et al.: Formal specification and verification of autonomous robotic systems: A survey. arXiv:1807.00048 (2018)

22. Kamali, M., et al.: Formal verification of autonomous vehicle platooning. Sci. of Comput. Program. 148, 88–106 (2017)

23. Karoui, O., et al.: Dual mode for vehicular platoon safety: Simulation and formal verification. Info. Sci. 402, 216–232 (2017)

24. Clarke, E.M., Henzinger, T.A., Veith, H.: Introduction to model checking. In: Handbook of Model Checking, pp. 1–26. Springer, Berlin, Heidelberg (2018)

25. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge, MA (2008)

26. Holzmann, G.J.: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, Boston, MA (2004)

27. Ben-Ari, M.: Principles of the Spin Model Checker. Springer, Berlin, Heidelberg (2008)

28. Shladover, S.E: Connected and automated vehicle systems: Introduction and overview. J. of Intel. Trans. Syst. 22(3), 190–200 (2018)

29. https://www.github.com/lost-vetern/project-immp

30. Ribeiro, B., et al.: Simulation and testing of a platooning management protocol implementation. In: International Conference on Wired/Wireless Internet Communication, pp. 174–185. Springer, Berlin, Heidelberg (2017)

31. Babiak, T., et al.: LTL to Büchi automata translation: Fast and more deterministic. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, pp. 95–109. Springer, Berlin, Heidelberg (2012)

How to cite this article: Fida, N.A., et al.: An improved multiple manoeuvre management protocol for platoon mobility in vehicular ad hoc networks. IET Intell. Transp. Syst. 15, 886–901 (2021).
https://doi.org/10.1049/itr2.12068