Fast Distributed Agreements and Safety-Critical Scenarios in VANETs

Gérard Le Lann

Fast Distributed Agreements and Safety-Critical Scenarios in VANETs

Gérard Le Lann
INRIA, BP 105, 78153 Le Chesnay Cedex, France

***Abstract*—We examine longitudinal and lateral safety-critical scenarios as they arise in VANETs, in the presence of concurrency and unreliable inter-vehicular communications. The Fast Distributed Agreement (FastDA) problem, as well as solutions to the longitudinal and the lateral instantiations of FastDA, are examined and informally specified. Analytical expressions of worst-case time bounds for reaching agreement are provided. We verify that stringent safety requirements are met through realistic examples drawn from various safety-critical scenarios.**

Keywords-Vehicular Ad Hoc Networks; Automated Vehicles; Time-Bounded and Reliable Inter-Vehicular Communications; Distributed Agreement; Safety.

I. INTRODUCTION

We consider ad hoc networks of autonomous or fully automated vehicles circulating on main roads and highways, a.k.a. VANETs (Vehicular Ad hoc Networks) [1]. Our focus is on safety-critical (SC) scenarios, where collisions are inevitable if not handled correctly. We examine longitudinal SC scenarios that may occur within a vehicular string and lateral SC maneuvers that span adjacent lanes/strings. Steep braking, velocity changes, shock waves (instability), are examples of the former. Lane changes, on-ramp merging, overtaking, lane merging, are examples of the latter. So far, with few exceptions, published work does not consider concurrency. In real settings, longitudinal or/and lateral SC scenarios may be undertaken simultaneously by multiple vehicles unaware of impending hazardous conflicts due to concurrency. For example, a vehicle V brakes abruptly, forcing its followers to decelerate steeply, thus reducing their inter-vehicular gaps, while another vehicle W attempts a lane change for being inserted between two V's followers, V not within sight of W. Being physically unfeasible, especially at medium/high velocities, that insertion must be prohibited. Another frequent scenario occurs when two vehicles, one circulating in the lane left of some lane j, another circulating in the lane right of j, undertake conflicting lane changes for being inserted in the same "slot" within a string in lane j.

We take the view that VANETs shall be analyzed as a collection of ad hoc strings—an isolated vehicle is a string of size 1—subject to conflicting behavioral "interferences", where safety may be jeopardized. Consequently, members of a set of nearby vehicles must agree explicitly on "what to do" prior to undertaking or granting SC maneuvers. This differs from current approaches where behaviors of autonomous vehicles rest on guessing possible moves of other vehicles, via robotics capabilities. Given that such guesses are not risk-free, large "safety margins" are mandatory, which is antagonistic with the various goals of "efficiency" targeted by autonomous driving. Also, this differs from classical dissemination schemes, whereby a single member (typically, a platoon leader) decides unilaterally to change some global parameter, e.g., velocity, other members having to instantiate this decision. Consequences of SC events received by other members while dissemination is underway are ignored.

Issues that arise with SC scenarios shall be addressed as problems in cyber-physics. The handling of a SC scenario rests on a pair $\{A, \Phi\}$, where A stands for a solution based on inter-vehicular communications, and Φ stands for control laws that govern vehicle behaviors in the physical space, according to decisions made via A. Execution of A shall prefix activations of Φ. (Processes Φ are not examined in this paper.) Only vehicle-to-vehicle (V2V) communications can be considered in A, since SC scenarios may develop far away from a road-side unit.

Our focus is on SC scenarios where the Fast Distributed Agreement (FastDA) problem arises. SC events instantiated as SC messages received by vehicles may be (i) concurrent, (ii) conflicting. Accidents can be avoided only if vehicles involved in a SC scenario select the same (unique) SC event, and behave accordingly. Agreement shall be reached in at most Ξ time units (a non-stochastic bound) under worst-case conditions regarding concurrency, message losses, number of vehicles involved, and MAC-level contention. Analytical expressions of Ξ must be given. Additionally, the STBA (Space-Time Bounds Acceptability) requirement shall be met: in Ξ time units, vehicles shall move by less than the average vehicle size (6 m approximately). It is well known that small values of Ξ cannot be achieved with V2V communication protocols defined in current IEEE or ETSI standards, under the aforementioned worst-case conditions. Novel communication protocols are needed.

In Section II, a system model is given and cohorts—a formalization of strings—are presented, along with the principles of neighbor-to-neighbor (N2N) short-range directional communications. Section III is devoted to LgFastDA, the longitudinal instantiation of FastDA, and to the Eligo agreement algorithm—Eligo is "I choose" in Latin. Section IV is devoted to LtFastDA, the lateral instantiation of FastDA which arises when lane changes are undertaken, and to the 3-phase LtHandshake protocol. In Section V, analytical expressions of bounds Ξ achieved by Eligo and LtHandshake are given. Bounds Ξ and distances travelled during these bounded delays are illustrated with some numerical examples.

II. SYSTEM MODEL

A. Introduction

Behaviors of vehicles are under the control of on-board (OB) systems, which include robotics devices (e.g., sensors, cameras, radars, lidars, actuators), radio communication devices, companion software, e-maps, and GNSS receivers (e.g., GPS, Galileo). Examples of functions available via robotics are lane-level positioning, safe longitudinal and lateral spacing. We assume lane numbering (consecutive integers), known via emaps. Robotics technology is limited to line-of-sight, and devices may fail, temporarily or permanently, thus the need for V2V communications. According to current standards such as IEEE 802.11p or ETSI ITS-G5, V2V communications rest on omnidirectional transmissions, radio range in the order of 250 m, interference range in the order of 400 m, 10 MHz channels, and a CSMA/CA-based MAC protocol [2].

Successful deliveries of V2V messages are not guaranteed with such standards [3], [4]. Furthermore, the problem of how to achieve small MAC-level non-stochastic bounds for channel access delays under realistic assumptions (e.g., numerous contenders, variable numbers of lanes, inaccurate GPS coordinates, message/reservation losses) remains unsolved. For example, see [5] where MAC delays achieved with the IEEE 802.11p protocol are evaluated resorting to analytical modeling and NS-2 simulations. For various channel loads, assuming 1 vehicle every 12 m, highest stochastic delays range between 75.3 and 211.8 ms. Exact worst-case delays are theoretically unbounded. That is not surprising. On a crowded highway (3 lanes each direction, 1 vehicle every 12.5 m), interference range in the order of 400m (as per IEEE 802.11p), the number of transmissions that may interfere with any given vehicle may be as high as 384. Assuming a V2V activity ratio in the order of 25%, up to 96 vehicles may attempt accessing a radio channel at the same time. Since achieving small values of Ξ is an elusive goal with current standards, other inter-vehicular communication protocols are needed.

Given that accidents involve vehicles necessarily sufficiently close to each other, short-range communications should suffice for the handling of SC scenarios. An interesting approach consists in addressing problems arising with lateral communications separately from problems as they arise with longitudinal communications in strings.

B. Cohorts and N2N Directional Communications

A cohort is an ad hoc string of vehicles, referred to as members, circulating in the same lane, bound to meet unambiguous specifications [6], [7]. To the exception of head *CH* and tail *CT*, every cohort member has two neighbors, separated by velocity dependent safe gaps denoted s_{xy} (see Fig. 1). A cohort circulating at velocity v may include up to $n^*(v)$ members, $n^*(v)$ being an inverse function of v, in compliance with the CSV (Cohort Size and Velocity) constraint: $v.n^*(v) < b_{csv}$, bound b_{csv} possibly stipulated in future standards. For example, with $b_{csv} = 2{,}200$ and v in km/h, one finds $n^*(108) = 20$, or $n^*(30) = 73$. Safe inter-cohort gaps (which depend on velocities), denoted $S_{ct/ch}$ in Fig. 1, are also specified. Thanks to these gaps, in case of a "brick wall" condition occurring in some cohort *C*, the head of a cohort following *C* never collides with *C*'s tail. Also, setting bounds such as $n^*(v)$ reduces the number of potential read-end collisions within a cohort (unbounded otherwise). Cohort-wide coordination can be accomplished via algorithms based on N2N directional communications. Short-range (e.g., 30 m), small beamwidth (e.g., 30°) front-looking and rear-looking directional radio antennas suffice [8]. N2N messages or beacons carry ranks. Members assign themselves consecutive ranks, 1 for *CH*, and $n \leq n^*$ for *CT*. An isolated vehicle assigns itself rank 1. A vehicle coming from behind would assign itself rank 2 upon receiving a beacon from its predecessor, and so on.

Cohort topology knowledge (CTK) is an important feature enabled by N2N communications. Periodically, string members generate N2N beacons carrying their respective intrinsic attributes (e.g., size and type), their ranks, and their current physical parameters (e.g., lane number, geolocation in lane, velocity, spacing with predecessor and successor). Beacons are propagated via a cohort-wide dissemination algorithm (see Subsection III.A). Thus, besides current n, every cohort member is knowledgeable of those intrinsic attributes and current parameters specific to other members, notably whether a member ranked r^{th} is located upstream or downstream relative to a given member's rank—see Subsection IV.A for how CTK can be used.

With short-range directional antennas, radio interferences may occur only among a limited set of nearby vehicles, which permits to solve the bounded channel access delay problem—see [9] for a detailed discussion of existing solutions. Here, a deterministic MAC protocol (SWIFT) specifically designed for strings [10] underlies our solutions to the FastDA problem.

C. Dependability Issues and Cohort Split

Returning an acknowledgement (ack) for every correct delivery of a N2N message is trivially feasible in cohorts, since neighbors know each other. Neighbors exchange beacons periodically, every π, whenever there is no messaging activity. Let p stand for a small integer, $p > 1$. A N2N link which experiences losses during $p\pi$ time units is declared failed. Failure of an OB system translates in a N2N link failure. With $p = 2$ and $\pi = 250$ ms, vehicles would travel less than 12.5 m at 90 km/h until a N2N link failure is detected. Safety is not sacrificed, since safe inter-neighbor gaps can be maintained via robotics capabilities. However, given our goal here, we must address the network partitioning issue. Most distributed agreement problems have no solutions in wired networks, in the presence of partitioning [11]. Such results do not hold with cohorts, for the following reason: *communication network partitioning leads to physical network partitioning*. This is called a cohort split. Consider cohort *C* and neighbors *X* and *Y*, *Y* following *X*. Assume a failed *X/Y* link. When *X* or *Y* detects that failure, that vehicle broadcasts a "cohort split" V2V message (to be received by all or some members of *C* and other cohorts), and activates a dissemination of this V2V message as a N2N message within the new cohort which is being created. *Y*

decelerates until safe spacing $S_{ct/ch}$ is instantiated between X and Y. X becomes tail of truncated cohort. When aware of the X/Y link failure, conditions permitting, X would accelerate so as to expedite the split maneuver.

For any fault-tolerant distributed algorithm, one must specify integer f, the highest number of (message, ack) losses that may be experienced in the course of execution. We can write $f \leq (p\text{-}1)\ (n\text{-}1)$ since within a split-free cohort, less than p consecutive losses may impact a N2N link.

In the sequel, cohort and string are used interchangeably.

D. SC Events/Messages and Instantiations of FastDA

LgFastDA is the agreement problem that arises within a string, or within a string subset, when membership remains unchanged. Consider a string C in some lane j. SC events are disseminated via N2N messages, to be processed by all or some members of C. Conversely, LtFastDA is the agreement problem that arises in SC scenarios where string membership is modified, due to the arrival of some external V2V message(s) sent from road-side clouds/units, or from vehicles external to C. Here, we examine the LtFastDA problem as it arises when some vehicle denoted R, referred to as a Requestor, wants to get inserted within C. In the sequel, the set of vehicles which must reach agreement is denoted Γ. With LgFastDA, Γ is string C or some C's subset, denoted C_0. With LtFastDA, Γ stands for $R \cup C_0$. Agreement is needed whenever members of Γ have to process concurrent conflicting SC events. The concept of conflict is defined and illustrated in Sections IV and V. Examples of SC events are "congestion ahead, velocity smaller than 30 km/h", "insertion in string C requested", which events may occur simultaneously.

SC scenarios and triggering events are assigned different types. In the presence of concurrent SC events, ties must be broken *uniformly*, i.e. identically by all vehicles involved. Tie-breaking between different types rests on priorities which depend on time or on road/traffic conditions. For example, a lane change of low priority may be prohibited or delayed, or be mandatory when tagged with a high priority (e.g., lane merging due to closed lane ahead). Due to space restrictions, use of priorities cannot be detailed in this paper.

III. LgFastDA and the Eligo Algorithm

A. Longitudinal FastDA vs. Message Dissemination

The presentation of LgFastDA is given for an entire string Γ. The merits of replacing V2V communications for intra-string coordination by N2N messages dissemination can be illustrated with velocity changes. Stability, avoidance of repeated amplifying accelerations and decelerations, is a major concern in platoons [12]. According to the CACC (Cooperative Adaptive Cruise Control) paradigm, a velocity change is triggered solely by a lead vehicle, such a change being broadcast via a V2V message, to be received and forwarded by platoon members. Given the weaknesses of V2V communications, authors have questioned the virtues of this approach. In analyses of CACC, V2V message losses are not always considered, and MAC access delays are often ignored, with some notable exceptions [13]. N2N message dissemination is essential for coping with losses. Consider 2 consecutive cohorts C_i and C_j, C_j following C_i, and imagine that an "emergency slow down" V2V message is broadcast by some member of C_i, which message is received by some members of C_j, to the exception of C_j's CH (CH is the leader in platoons). Only these members are able to launch the desired deceleration process within C_j. This is just one example showing that it is necessary to address issues arising with SC scenarios assuming that any string member may be the triggering vehicle, and that dissemination may proceed upstream, in addition to downstream.

Dissemination is mandatory, since achieving agreement implies that all messages which may carry different SC events are delivered to every vehicle involved in a SC scenario. In [10], building on [14], we have presented SWIFT, a MAC protocol which achieves fast string-wide reliable message dissemination. Worst-case time bounds Ξ (agreement) are derived from bounds Δ achieved by SWIFT (dissemination).

B. Informal Specification of LgFastDA

Approximate agreement and exact agreement, a.k.a. consensus, are among the most studied algorithmic problems in distributed computing [11], [15]. However, neither the problem specifications nor the solutions devised for models of wired systems are applicable to cyber-physical systems such as VANETs. Consensus (see below) is a good example.

Assumptions
- Asynchronous or synchronous system model.
- Set of n processes, $n > 1$.
- Less than n incorrect (failed) processes.
- Reliable inter-process communications.
- Every process proposes a value.

Properties
- *Validity*: Decision value is some value proposed.
- *Agreement*: No two correct processes decide differently.
- *Eventual Termination*: Every correct process eventually decides.

LgFastDA differs from Consensus. Firstly, regarding *Validity*, a very specific decision value D ought to be chosen among proposed values, since D must match physical constraints such as geolocations and velocities. For example, in the case of on-ramp merging, which pair of vehicles is to be chosen for insertion cannot be "some" pair on a highway. Secondly, *Eventual Termination* is replaced by *Time-Bounded Termination*, a stronger requirement. Thirdly, the *STB Acceptability* and the *Synchronicity* requirements do not appear in traditional specifications of Consensus. Fourthly, we assume unreliable communications, and we explicitly address the issue of how to cope with losses. We have seen that failures of N2N links lead to cohort splits (process failures are our OB system failures). Consequently, N2N communication failures can be ignored in the design of algorithms that solve LgFastDA, since this problem is defined for a non-disconnected cohort. If being run when a cohort split occurs, an agreement algorithm is aborted, and restarted within newly formed cohorts.

TABLE I. INFORMAL SPECIFICATION OF LGFASTDA

■ Assumptions
- Synchronous system model.
- String Γ of n contiguous members, $n > 1$, head denoted *CH*, tail denoted *CT*.
- OB system failures and N2N link failures either are tolerated or result in a string split.
- Every member may propose a value.

■ Properties
- *Validity*: Decision value $D = \Psi\{\text{proposed values}\}$.
- *Agreement:* No two members decide differently.
- *Time-Bounded Termination*: Every member decides in at most Ξ time units.
- *STB Acceptability*: Distances travelled in Ξ are smaller than average vehicle size.
- *Synchronicity*: Times at which *D* is posted to OB systems are comprised within a small time interval ε. Distance travelled during ε by the member earliest to post *D* until the latest member does so is an order of magnitude smaller than vehicle sizes.

C. Solving LgFastDA—The Eligo Algorithm

The informal specification of Eligo is given in Table II. Values relative to a given type (see Subsection II.D) are drawn from a set associated to that type. Solving LgFastDA is relatively easy from an algorithmic standpoint. The challenge of interest is to find an efficient solution, leading to smallest best-case and worst-case termination times. Eligo is a 2-phase algorithm. SWIFT is used for downstream and upstream dissemination of N2N messages throughout Γ (see Fig. 1). Names of N2N messages are init, collect, and decisive, underlined in the presentation below.

1) Overview

Let v_k stand for a value proposed by member *K*, referred to as a proposer. *D* is any appropriate function Ψ of proposed values, e.g., smallest value, relative majority, Boolean disjunction. The Agreement requirement is met by providing every member with common knowledge (all proposals or *D*).

a) Notations and variables

- init: sent in phase 1
- collect and decisive: sent in phase 2
- [message]: contents of message
- *T**: termination time (UTC) of Eligo
- *u*: time needed for computing *D*
- Timer *TT*: time left until reaching UTC time *T**
- Boolean *P* (true ≡ I have a proposal)
- Boolean *FI* (true ≡ I have forwarded init before)
- $D = \Psi\{[1^{st}\ \text{collect}] \cup [2^{nd}\ \text{collect}]\}$

b) States

- *listen*: state entered upon termination of Eligo; entering this state resets Booleans *P* and *FI* to false.
- *prop*: state entered after a collect message has been forwarded, or after creating a collect message (the latter done by *CH* or *CT*).
- *waitT*: OB system is inactive or runs algorithms other than Eligo, until timer *TT* awakes.

c) Events

e_1: issuance of proposal as requested by local OB system, in response to some external event or internal state transition
e_2: init received from a neighbor
e_3: collect received from a neighbor
e_4: decisive received from a neighbor
e_5: awakening of timer *TT*.

Key to achieving smallest best-case termination bounds is the choice of sending init messages in both directions during phase 1.

2) The Eligo algorithm

Phase 1 starts when some member sends an init message to its neighbor(s), message forwarded upstream and downstream so as to awake *CH* and *CT*, in case neither *CH* nor *CT* awakes first. The specification in Table II is given for some member *Y* other than *CH* and *CT*, assuming $n > 2$ (the case $n = 2$ derives trivially from Table II). Upon entering phase 1, a member is in state *listen*. To the possible exception of *CH* or *CT*, a member must have issued or relayed an init message prior to receiving a collect message (e_3), which triggers switching to state *prop*. A member sends or forwards an init message only once. Only 1 proposal may be issued by a member while Eligo is running. There is no phase 1 when *CH* or *CT* awakes spontaneously. Phase 2, started by *CH* (resp., *CT*), or by both of them concurrently, consists in disseminating a collect message that carries all proposals from visited members until reaching *CT* (resp., *CH*), or until crossing another collect message (*W* in Fig. 1). The member where this crossing occurs knows all proposals.

TABLE II. INFORMAL SPECIFICATION OF ELIGO

State/Event Transitions for $Y \neq CH$, $Y \neq CT$

listen ⊗ e_1: **if** *P* false **then** {v_y:= proposal; *P*:= true; send init to both neighbors} **else** {**if** new proposal of priority higher than proposal recorded in v_y **then** v_y:= new proposal}
listen ⊗ e_2: **if** *FI* false **then** {forward init to opposite neighbor; *FI*:= true} **else** discard init
listen ⊗ e_3: *%1st collect received%*
if *P* true **then** {add v_y to [collect]}; store [collect]; forward collect to opposite neighbor; switch to *prop*
prop ⊗ e_3: *%2nd collect received%*
compute $D = \Psi\{\text{values in both collect}\}$; compute *T** and *δT*; store *D* and *T** in decisive; forward decisive to both neighbors; set timer *TT* to *δT*; switch to *waitT*
prop ⊗ e_4: forward decisive to opposite neighbor; compute *δT*; set timer *TT* to *δT*; switch to *waitT*
waitT ⊗ e_5: post *D* to OB system; switch to *listen*
prop ⊗ e_1: % Eligo is in progress%
new proposal put on hold
waitT ⊗ e_1: % Eligo is in progress%
new proposal put on hold

Therefore, D can be computed. In case two neighbors are provided each with both collect messages, both would compute the same D. When a collect message is created by CH or CT at UTC time t, termination time $T = t+u+\Xi$ is computed and recorded in the collect message. (Bound Ξ is detailed in Subsection V.A.1.) Ignoring discrepancies ε relative to UTC readings (see Synchronicity), CH and CT compute the same T, except when string membership is being updated, in which case CH and CT may use different values for n, thus may compute 2 different times T and T'. In order to avoid any ambiguity, the member which computes D also computes time $T^* = \max\{T, T'\}$. D and T^* are recorded in the decisive message. A member that is delivered a decisive message at UTC time t_d computes $\delta T = T^* - t_d$, and sets its timer TT to δT. All members instantiate D at the same UTC time (see state transition $waitT \otimes e_5$).

As long as state *prop* is not entered, new local proposals are accepted. In case a waiting queue would build up, the proposal of highest priority is stored in the first collect message seen. Proposals put on hold are serviced by reactivating Eligo once the current instantiation is over. Members do not defer the relaying or the processing of N2N messages unrelated to Eligo while Eligo is running.

It is easy to check that the *Validity* and the *Agreement* properties hold. (Correctness proofs are by contradiction.)

IV. LTFASTDA AND THE LTHANDSHAKE PROTOCOL

We examine LtFastDA in the following setting: a vehicle referred to as a Requestor, denoted R, which circulates on a lane j' adjacent to lane j, wants to be inserted between 2 members of C_0, subset of string C in lane j. Thus, $\Gamma = R \cup C_0$. Solving LtFastDA in Γ implies showing how C_0 is identified (agreement is reached with Eligo run in C_0). Once agreement has been reached, R is returned a response which can be negative (lane change is denied), or positive, in which case the V2V messages sent to R carry the data needed for performing a safe lane change. Assumptions and properties given in Table I apply fully here, with $\Gamma = R \cup C_0$, n replaced by g, CH replaced by F (first) and CT replaced by L (last).

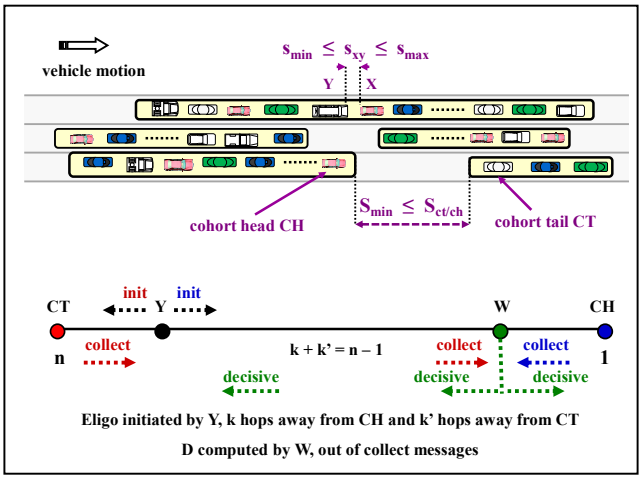

Figure 1. Cohorts and principles of Eligo illustrated ($Y \neq CH$, $Y \neq CT$)

A. Solving LtFastDA—The LtHandshake protocol

The Lateral Handshake (LtHandshake) protocol is composed of 3 phases, shown in Table III. In phases 1 and 3, vehicles exchange V2V messages via Lateral Geocast (LtGcast) and Unicast (Ucast) primitives, respectively. Eligo is run in phase 2. In phase 1, R only needs to "talk to" nearby vehicles in lane j. Therefore, LtGcast—Geocast [16] aimed at laterally positioned vehicles, suffices. The problem of achieving fast V2V message deliveries with omnidirectional communications is still open. Consequently, we assume the following (hypothesis H):

- When LtGcast$\{M,V\}$ is performed by vehicle V, at least 1 vehicle among those targeted by V is delivered message M correctly and in time,
- When several Ucast$\{M,V\}$ are performed directed at the same vehicle V, V is delivered at least 1 message M correctly and in time.

In our system model, a vehicle only needs to know its geolocation *in its lane*, with a precision better than average vehicle size. In addition to data provided by GNSS receivers, which can be inaccurate at times (loss of satellite signals), the necessary precision can be obtained via dead reckoning, i.e. distances measured to or from a landmark (found in emaps). Nearby vehicles in adjacent lanes have comparable geolocations in their respective lanes.

In phase 1, R does LtGcast$\{Q,R\}$, Q standing for a V2V lane change request message, identity $id(Q)$, which carries parameters $\varphi(R)$ such as velocity, size, and geolocation in j'. Let C_0 stand for a group of g contiguous members in C which match $\varphi(R)$, referred to as Participants. Most often, g is not higher than 5 (4 possible insertion slots). LC*test* stands for a procedure which identifies C_0 from CTK and $\varphi(R)$, or which returns "nil" (not a Participant). Very briefly, in LCtest run by Y, a member of C, Y's and R's geolocations are compared against each other, as well as whether spacing with Y's predecessor and/or successor need be increased, for accommodating R's insertion.

In phase 2, Eligo is run by Participants. Proposed values carry the identity $id(.)$ of the request being answered. More precisely, Y proposes "OK for lane change $id(Q)$, my value is v_y = [Y's size, geolocation in j, spacing $s_{y,x}$ (resp., $s_{y,z}$) with predecessor (resp., successor)]". There are various ways of computing D, ranging from simple and sub-optimal to elaborate. Here is an example of the latter with triple (X,Y,Z), where Y precedes Z and follows X. Let $d_R(V,W)$ stand for the distance to be covered by R for an insertion between V and W, and $\delta_R(V,W)$ stand for the additional spacing to be created between V and W for R's insertion. If the objective is to minimize overall energy consumption due to accelerations and decelerations, then function $\Psi = \min\{d_R(V,W)+\delta_R(V,W)\}$ computed over C_0 would make sense. Let $D(R)$ stand for the agreed decision value. $D(R)$ is a pair of "elected" neighbors (names and related data, such as, e.g. geolocations in j). See Subsection V.C for conflicting concurrent lane changes.

In phase 3, members of C_0 do Ucast$\{id(Q),D(R),R\}$. Note that all Ucast messages carry the same $D(R)$. Elected vehicles create an insertion slot for R and R starts moving toward that slot upon receiving a Ucast message.

TABLE III. THE LTHANDSHAKE PROTOCOL

- Phase 1: R does LtGcast$\{Q,R\}$.
- Phase 2: Upon receiving Q from R, Y runs LC*test*. If "nil" is returned, no further phases. If a Participant, Y runs Eligo, proposal v_y.
- Phase 3: When Eligo delivers decision $D(R)$, Y does Ucast$\{id(Q),D(R),R\}$.

Under H, since R and members of C_0 have seen every V2V message exchanged in phases 1 and 3, R and members of C_0 are in agreement. Note that $D(R)$ could be computed by R, relieving Participants from the need to run Eligo. However, since Ucast messages would not have identical contents (individual proposals differ), the loss of a Ucast message in phase 2 could lead to an incorrect decision, even under hypothesis H. Moreover, that would entail a third round of V2V messaging, since R must "tell" Participants which of them are elected, correctly or incorrectly.

B. Conflicting Concurrent Lane Changes

Within a given string, multiple insertions can be performed quasi simultaneously provided that they do not conflict, i.e. they are physically feasible and risk-free. With Requestors sufficiently apart from each other, concurrency is conflict-free. Conversely, consider 2 lane changes requested at about the same time by requestors V and W via LtGcast$\{P,V\}$ and LtGcast$\{Q,W\}$. V and W want to move to lane j. In one scenario, both of them are in lane j-1. In another scenario, V is in lane j-1, W in lane j+1, V and W not in mutual line-of-sight. Let write $C_0^* = C_0(P) \cap C_0(Q)$.

There is no conflict when $C_0^* = \varnothing$. A conflict arises whenever $C_0^* \neq \varnothing$. At least 1 member of C_0^* has issued a proposal "OK for lane change $id(P)$" and at least 1 member of C_0^* has issued a proposal "OK for lane change $id(Q)$". Safety requires 1 acceptance at most in case of a conflict, which can be trivially enforced: requests are systematically rejected, lane changes prohibited. Such is the case with existing autonomous vehicles: non-conflicting lane change attempts may fail, due to large "safety margins". Arriving at such an unsatisfactory outcome can be avoided with Eligo. Ties are broken deterministically, for example by defining some total order (ω) on the set of identities $id(.)$, and by resorting to a choice function (Ψ) relative to ω (e.g., $\min_\omega$ or $\max_\omega$), leading to a unique id^* over C_0^*. Values retained as inputs for Ψ are those tagged with id^*. It follows that calculations of $D(.)$ are relative to a unique request. For example, with $\max_\omega$ as the choice function, W would change lane first if $id(Q) >_\omega id(P)$, followed by V afterwards.

V. TIME BOUNDS AND PROPERTIES ACHIEVED

Let θ stand for the transmission duration of the largest N2N message, receiver's OB processing time included. Beacons and acks being shorter than N2N messages, their N2N link delays are smaller than θ. SWIFT achieves the following string-wide worst-case dissemination time [10]:

$$\Delta(n,f) \leq 2h\theta \ \{f+1+\lceil (n-1)/h \rceil\},$$

where $2h\theta$ is the worst-case channel access delay. Realistic highest values of integer h are 3 or 4.

Let $\Xi_{LgDA}(x,f)$ stand for the worst-case termination time of Eligo when run in a string of x members, $x = n$ (string C) or $x = g$, the number of members in C_0. Let $\Xi_{LtDA}(g,f)$ stand for the worst-case termination time of LtHandshake.

A. Time-Bounded Termination

1) Eligo

Worst-case $\Xi_{LgDA}(n,f)$ is experienced whenever Eligo is initiated by *CH or CT*, since reaching agreement in this case necessitates 2 consecutive disseminations throughout C, in opposite directions. Consequently, we have:

$$\Xi_{LgDA}(n,f) \leq 2h\theta \ \{f+1+ 2\lceil (n-1)/h \rceil\}. \quad (1)$$

Trivially, in C_0, we have:

$$\Xi_{LgDA}(g,f) \leq 2h\theta \ \{f+1+ 2\lceil (g-1)/h \rceil\}. \quad (2)$$

Bound $\Xi_{LgDA}(n,f)$ depends on n and f. Integers n and f are common knowledge, since the current value of n is known to every member through CTK (see Subsection II.B), and the current value of f may be updated at will if so desired, propagated via SWIFT. Therefore, all members use the same Ξ_{LgDA} whenever Eligo is started. Smallest termination times are achieved when Eligo is initiated by a mid-point member, in which case $2\lceil (x-1)/h \rceil$ would be replaced by $3/2\lceil (x-1)/h \rceil\}$ in Eq. (1) and (2). Recall that these bounds hold whatever the number of nearby V2V transmitters (specific radio channels are allocated to short-range N2N communications).

2) LtHandshake

Let σ_{max} stand for the worst-case latency involved with transmitting and delivering a V2V message, MAC access delay included (phases 1 and 3). Compared to σ_{max}, OB system latencies due to running the code of Eligo are negligible, thus ignored here. Under H, losses of V2V messages can be ignored. Therefore:

$$\Xi_{LtDA}(g,f) \leq 2\,\sigma_{max} + \Xi_{LgDA}(g,f).$$

The key parameter is σ_{max} which, strictly speaking, may take unbounded values due to contention and message losses. Let us now revisit hypothesis H. To the best of our knowledge, there are no solutions to the lane change problem in case H would be repeatedly violated when LtGcast is performed. This can be tolerated by resorting to SWIFT, under a more permissive hypothesis: at least 1 member of C (say Z) not a member of C_0 receives a V2V message issued via LtGcast. Recall that F (resp., L) stands for C_0's member of smallest (resp., highest) rank. Let z-1 ($z > 2$) stand for the hop distance between Z and F or L. Dissemination of such a message from Z to F or L is then needed for reaching C_0, which entails the following additional latency:

$$\Delta(z,f_d) \leq 2h\theta \ \{f_d+1+\lceil (z-1)/h \rceil\},$$

where f_d stands for the number of N2N message/ack losses experienced during dissemination.

3) Numerical assessment of time bounds

Time intervals or bounds are in ms, v in km/h. Consider a N2N channel bandwidth in the order of 15 Mbits/s and N2N messages smaller than 5 Kbits. Values of θ would range between 0.3 and 1.2. Let us choose $\theta = 1$, and $h = 4$ (a conservative value).

For Eligo run in cohort C, consider the following cases: $\{n = 20, f = 6\}$ and $\{n = 88, f = 21\}$. One finds:

$\Xi_{LgDA}(20,6) \leq 136$, and $\Xi_{LgDA}(88,21) \leq 528$.

For Eligo run in C_0, let us choose $g = 5$ and $f = 1$. For LtHandshake under H, one finds: $\Xi_{LtDA}(5,1) \leq 32 + 2\,\sigma_{max}$.

When H does not hold, observe that Z cannot be too far away from R, thus from F or L. Let us have $z = 3$, and $f_d = 1$. The additional dissemination delay is $\Delta(3,1) \leq 24$. As expected, additional latencies due to relaxing hypothesis H are quite small. Dissemination as per SWIFT improves dependability in SC scenarios that involve loss-prone V2V messaging, without sacrificing timeliness.

B. Other Properties

1) STB Acceptability

According to the CSV constraint, we have $v.n^*(v) < b_{csv}$. Let us choose $b_{csv} = 2{,}200$ (see Subsection II.B). Thus, with those cases considered above, we have $v < 110$ for $n = 20$, and $v < 25$ for $n = 88$. Therefore:

- During $\Xi_{LgDA}(20,6)$, distance covered < 4.16 m,
- During $\Xi_{LgDA}(88,21)$, distance covered < 3.67 m.

Eligo meets the STB Acceptability requirement.

During $\Xi_{LtDA}(5,1) + \Delta(3,1) - 2\sigma_{max}$, at $v = 110$ (most pessimistic value), distance covered < 1.71 m. Whether LtHandshake meets the STB Acceptability requirement depends fully on σ_{max}.

2) Synchronicity

OB systems activate processes Φ when local timers TT awake, at the same UTC time T^*, inaccuracy ε. If needed, OB GNSS receivers may be backed up by OB clocks. Affordable clocks with accuracy and stability figures in the order of $1x10^{-6}$ achieve ε in the order of 1 µsec/second, which permits to cope with 1 minute long GNSS outages. Discrepancies in the reading of UTC time by any 2 vehicles would then be in the order of 120 µsec, i.e. $3.6x10^{-3}$ m at 108 km/h. The Synchronicity requirement is met.

C. Generalized Conflicting Concurrency

An example of conflicts due to SC events of the same type has been examined in Subsection IV.B. An example of conflicts arising with SC events of different types has been given in the Introduction. Another example in a multilane highway would be as follows: SC event "clear lane j" issued by an ambulance, SC event "steep braking" aimed at string C in lane j, SC event "request from V for lateral insertion in C in lane j". Assuming that an emergency V2V message is broadcast by the ambulance does not suffice. If "clear lane j" is of highest priority, then all members of C must decide to move to adjacent lanes (V shall not leave its lane). We have seen that Eligo handles such scenarios, within latencies such that vehicles do not move much by the time they know what to do. The CSV constraint, which is intrinsically linked to the cohort concept, is fundamental in this respect. Moreover, these emergency lane changes must be risk-free. This implies that members of strings in lanes adjacent to j must also agree on how to accommodate such unexpected insertions, safely and quickly. We have just brought to light a complex SC scenario which, so far, has not been thoroughly investigated.

VI. PERSPECTIVES

The future of autonomous/automated driving depends on whether safety issues are addressed rigorously. In this paper, we show that it is possible to design distributed deterministic decision-making algorithms of very low complexity, hence of very small bounded execution delays. It is thus possible to contemplate the prefixing of risk-prone physical maneuvers with such algorithms (pair $\{A, \Phi\}$), since vehicles cover very small distances by the time they know what to do, safely. Numerous algorithmic problems related to safety have not been addressed yet. They should be.

REFERENCES

[1] G. Karagiannis et al., "Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions", IEEE Comm. Surveys & Tutorials, vol. 13, 4, 4th quarter 2011.

[2] M.L. Sichitiu and M. Kihl, "Inter-vehicle communication systems: A survey", IEEE Comm. Surveys & Tutorials, vol. 10, 2, 2008, pp. 88-105.

[3] C. Bergenhem et al., "V2V communication quality: Measurements in a cooperative automotive platooning application", SAE Intl. J. Passeng. Cars – Electron. Elec. Syst., vol. 7, 2, Aug. 2014, 9 p.

[4] K. Karlsson, C. Bergenhem, and E. Hedin, "Field measurements of IEEE 802.11p communication in NLOS environments for a platooning application", IEEE VTC Fall-2012, pp. 1-5.

[5] Y. Yao et al., "Delay analysis and study of IEEE 802.11p based DSRC safety communication in a highway environment", Proc. IEEE Infocom 2013, pp. 1591-1599.

[6] G. Le Lann, "Cohorts and groups for safe and efficient autonomous driving on highways", 3rd IEEE Vehicular Networking Conference (VNC), Amsterdam (NL), Nov. 2011, pp. 1-8.

[7] G. Le Lann, "Integrated Safety and Efficiency in Intelligent Vehicular Networks: Issues and Novel Constructs", Proc. TRA 2012 - Transport Research Arena Europe, Athens, Greece, ScienceDirect, Elsevier, 2012, vol. 48, p. 951-961.

[8] R. Ramanathan et al., "Ad hoc networking with directional antennas: A complete system solution", IEEE Journal Selected Areas in Communications, vol. 23, 3, March 2005, pp. 496-506.

[9] A.A. Abdullah, L. Cai, and F. Gebali, "DSDMAC: dual sensing directional MAC protocol for ad hoc networks with directional antennas", IEEE Trans. Vehicular Technology, vol. 61, 3, March 2012, pp. 1266-1275.

[10] G. Le Lann, "A collision-free MAC protocol for fast message dissemination in vehicular strings", IEEE CSCN'16 Conference, Berlin, Oct.-Nov. 2016, 7 p.

[11] N.A. Lynch, Distributed Algorithms. Morgan Kaufmann. ISBN 1-55860-348-4 (1996), 872 p.

[12] G.J.L. Naus et al., "String-stable CACC design and experimental validation: A frequency-domain approach", IEEE Trans. Vehicular Technology, vol. 59, 9, Nov. 2010, pp. 4268-4279.

[13] C. Lei, and al., "Impact of packet loss on CACC string stability performance", 11th Intl. Conference on ITS Telecommunications (ITST 2011), Aug. 2011, pp. 381-386.

[14] G. Le Lann, "Safety in vehicular networks—On the inevitability of short-range directional communications", Proc. 14th Intl. Conference on Ad Hoc, Mobile, and Wireless Networks (AdHoc-Now 2015), Athens, June-July 2015, Springer LNCS 9143, pp. 347-360.

[15] M. Biely, U. Schmid, and B. Weiss, "Synchronous consensus under hybrid process and link failures", Theoretical Computer Science, 412(40), 2011, Elsevier, pp. 5602–5630.

[16] R. J. Hall, "An Improved Geocast for Mobile Ad Hoc Networks", IEEE Trans. Mobile Computing, vol. 10, 2, Feb. 2011, pp. 254-266.