A New Token-based GME Algorithm for the Distributed AIM

Sung Hoon Park[1], Bo Kyong Kim, Young Chul Seo and Yeong Mok Kim[2]

School of Eletronical and Computer Engineering,

[1]Chungbuk National University, Korea

[2]spark@cbnu.ac.kr, yeongmokkim@gmail.com

Abstract *– According to the development of sensor technology and ubiquitous networking, autonomous vehicles(AVs) are moving around the road and will be ready for public use in the near future. With the advent of such AVs, the concept of Autonomous Intersection Management (AIM) without traffic light has attracted great attention over the last decade and will be a promising option and various researches related are being done actively. To solve the problems of intersection control unit based system, AIM based on the distributed system should be developed. However to our knowledge there is almost no work that has been devoted to AIM based on the distributed system.*

This paper proposes a token-based group mutual algorithm for the distributed AIM. We use a token as a privilege to pass cross zone(critical section in an intersection) and use one token. How to transfer the token among vehicles in a dynamic system in which a vehicle can not stay in the cross zone after it finish passing it is the core of our algorithm. We design the circulation of the token efficiently among vehicles and session declare concept to improve the flow of vehicles.

The proposed algorithm can handle quite a big traffic volume well with decreased message complexity. It outperforms the reference algorithms in message complexity and presents better result in system throughput than traffic signal system. This paper also will vitalize the researches about the distributed AIM.

Keywords: Token-based, GME, AIM, distributed

I. Introduction

According to the development of sensor technology and ubiquitous networking, autonomous vehicles(AVs) are moving around the road and will be ready for public use in the near future. With the advent of such AVs, the concept of Autonomous Intersection Management (AIM) without traffic light has attracted great attention over the last decade and will be a promising option. In the scheme of AIM, AVs communicate with the intersection controller to exchange the information of their states. Also, AVs are commanded and coordinated by an Intersection Control Unit (ICU) to cross the intersection safely, as the vehicles can control their states accurately by the sensors mounted in them. Live AV data are collected by sensors and sent to the ICU in real time. The right-of-way is assigned to each AV according to its state and the state of the intersection. Only AVs that have the right-of-way can pass through the intersection. AIM aims at calculating the optimized trajectory and determining the intersection passing sequence with the collected information for which the ICU will continue to share information with nearby AVs. However, to implement the AIM scheme which is based on the ICU as a central control equipment in the intersection, we may face big problems as follows:

First, the scheme requires ICU installation work which is costly and once installed, the design is not flexible (i.e. it is not easy to change or improve).

Second, the serious problem is that if an ICU fails, the entire AIM system will not work and can cause tremendous disruption.

In order to solve these problems of the existing AIM, the scheme of AIM should be developed based on the distributed computing system. In other words, in order for an AV to safely pass through an intersection without an ICU, it is necessary to thoroughly collaborate through real-time information exchanged between AVs.

However, to the best of our knowledge there have been no studies on the decentralized distributed systems for AIM(D_AIM) except the algorithm of Weigang, Jiebin (Distributed Mutual Exclusion Algorithm for Intersection traffic control, 2015) and our earlier work(*VTokenIC*, 2017).

In this paper, we design an efficient token-based algorithm for D_AIM that is especially suitable for applications in which group selection probability may be non-uniformly distributed. There is only one token in the intersection and the system proposed consists of continuous flow of sessions.

The rest of this paper is arranged as follows. In section 2, we describe the system model assumed and detailed environmental factors of the system. In section 3, we present the proposed algorithm for the distributed AIM. In section 4, we explain performance analyses of the proposed algorithm. Finally in section 5, we give concluding remarks.

II. System model and problem definition

2.1 System Model

We assume an asynchronous message-passing distributed system comprising a set of *n* AVs. Generally, AVs are to continuously arrive at an intersection and leave it and the number of AVs in the system changes all the times. However, at a particular time, there are only fixed number of AVs that are moving in the system. Also, the traffic flow can be truncated into segments by lanes, which simplifies the problem. At that time, there are *n* AVs, AV = {AV_1, AV_2, ..., AV_n} in the control range (i.e. system) approaching an intersection from different approaches and passing through the intersection; the essential data of all AVs will be communicated with each other by sending messages over a set of channels.

The mobile ad hoc network model is defined as an undirected graph, G = (V, E). Each vertex of a set of vertices, V = {v_1, v_2, ..., v_n} (n>=1), represents a mobile node.

Information exchange between AVs is done by asynchronous message passing. Further, we assume each vehicle has a unique ID and all the channel are reliable and a network failure does not occur.

2.2 Group mutual exclusion problem

In a traditional mutual exclusion theory, only one process can access the critical section at any moment. However, the intersection control problem in our algorithm proposed is based on group mutual exclusion theory. AVs at a same group can access the critical section (i.e. pass the intersection as a group) simultaneously.

The GME problem in the algorithm proposed is to hold the following properties;

(1) *Safety*

If two or more AVs are in the critical section at the same time, then they are in a same group.

(2) *Liveliness*

AVs that enters the system and requests intersection passing must enter the intersection of the system.

(3) *Concurrent entering*

If requests of all AVs belong to the same group, they must not be asked to wait until another AV leaves the intersection.

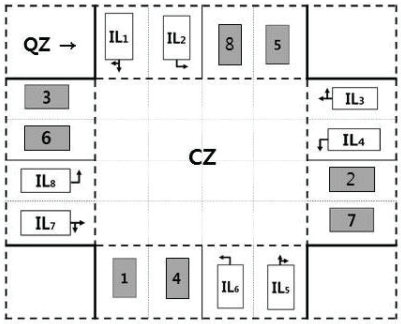
2.3 intersection

As shown in Fig. 1, we assume an intersection with four directions, i.e. north, south, east and west. In each direction, there are two lanes. Typically, an intersection(IC) consists of a number of approaches and a crossing zone(CZ). Each approach may be used by several traffic streams. For example, for the IC in Fig. 1, the approach from west to east consist s of two traffic streams (IL 3 and 4). Each stream has its own lane and an independent vehicle queue. The path used by a traffic stream to traverse the IC is called the trajectory. A trajectory connects an approach on which AVs enter the IC to the IC leg on which these AVs leave the IC.

The objective of AIM is to transform input traffic flows into output ones while preventing traffic conflicts and satisfying a specific criterion.

(*Fei Yan. 2013, an autonomous vehicle sequencing problem at intersections*)

The dashed range represents the IC and all AVs inside the range will be considered as a member of the system to take part in cooperative driving; whereas AVs outside of the circle will not be considered temporarily. The radius of this virtual range should be determined appropriately by inter-vehicle communication protocol that has been selected for this application.



<Figure 1> Intersection

- *definition 1*: compatible and conflict relationship

When trajectories of two traffic streams do not cross, these streams can simultaneously get the right-of-way and we call these two streams compatible stream. The lanes on which the two streams are moving are called compatible lanes. On the other hand, when trajectories of two traffic streams do cross, the streams are in a conflict relation.

- *definition 2*: compatible stream group (CSG)

When several stream are compatible with each other, we call the set of these streams a compatible stream groups (CSG). In the system proposed, we can partition the eight streams into four CSGs as below:

(CSG 1 – L1 & L5), (CSG 2 - L2 & L6),

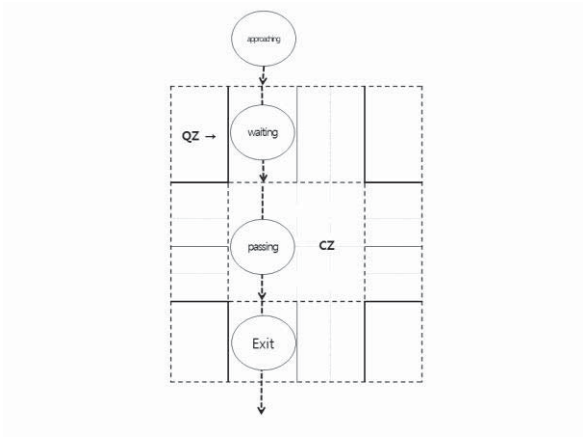
(CSG 3 – L3 & L7), (CSG 4 – L4 & L8)

- *definition 3*: vehicle passing group (VPG)

A large number of AVs may be waiting in the lane to pass through the CZ. If each AV act and enter the crossing area individually, the performance of the system will be decreased. We therefore define VPG as a set of vehicles from the same stream that pass the CZ without the interruption of the other AVs in the IC. Eight VPGs can be generated in the IC based on each traffic stream. AVs in stream 1 is VPG 1, e.g. AVs in stream i is VPG i. Each VPG consists of AVs in the same lane and the head and tail of the VPG is decided based on the arrival time. Each AV should be in one and only one VPG.

2.4 The state of AV_i

The movement of AV_i passing through IC can be described by an automaton with three states, as shown Figure 2.



<Figure 2> the state of AV_i

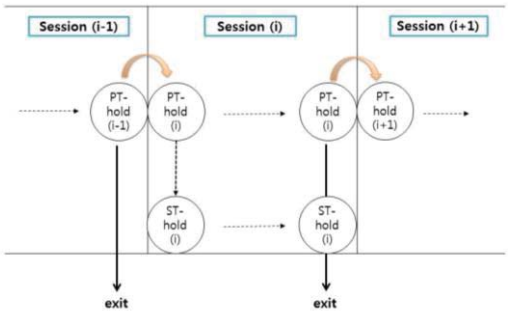
• *idle*
At this step, as AV_i is out of the IC, it is not controlled by the system.
• *waiting* state
The state is that AV_i waits for the CZ passing and AV_i is in the state from when entering the IC to when entering the CZ.
·*passing* state
The state is that an AV moves to pass the CZ. AV_i will be in the state during the time interval between entering the CZ and exiting it.

III. The proposed algorithm - *TDGim*

In this section, we describe a token-based GME algorithm for the autonomous intersection management (AIM) and we call it *TDGim*. *TDGim* is a distributed AIM in which all AVs in the system collaborate with each other only through message exchanges between vehicles and traverse through the intersection safely without a traffic light.

3.1 the main idea

TDGim is a token-based GME algorithm in which the token holder manage a session. The token mentioned here means a message with special authority and holding the token means that it is qualified to pass through IC. A single token is used for the system and continuously rotated. The token has a group of the session associated with it and it can only be used to enter the CZ of IC of that type. The session is time interval during the token holder and the all AVs in the same VPG with the token holder enter the CZ and pass through it.
An AV, if it wants to access a session, on receiving the token, initiates the session and also declares the session to all other AVs in the IC. Any AV that listens to this session can enter the CS for the same session concurrently. In other words, an AV, when it wants to access the CZ for a particular session, checks in the entry section whether the session is available. If it is, the AV can join the session concurrently. On the other hand, if an AV wants to access other session, it sends a request to the token holder for the token. After the current session is over, the current token holder sends the token to the requesting AV.
Thus, *TDGim* is the algorithm for a continuous session flow in which the token holder is changed and thereby the group of AVs to be entered the CZ is changed. (shown in Figure 3)



<Figure 3> Session Flow

3.2 description of *TDGim*

3.2.1 *Waiting section*

AV_i approaches to the IC if it is interested in the CZ entering. AV_i is in *Waiting section* when it is located inside the circle of the IC. AV_i in *Waiting section* should wait for the CZ entry in the lane and it may receive the *token* or the session declare message or a notify message.

(1) receiving a session declare
AV_i, on receiving a session declare from the *key AV* in *Waiting Section*, updates the variables associated with the session group and ID of the *key AV* and should check the value of the declared session. There are two cases depending on the group of the session as follows.
①if the value of group of the session declared is same with that of AV_i, it can join the session concurrently.
②if it is not, AV_i should send a *request* message to the *key AV*

for the *token*. The request message contains ID of AV, the value of the session requiring and the current time.

(2)receiving a *notify* message
AV_i, on receiving a *notify* message from the temporal token holder, if it did not receive a *declare* message or another *notify* message yet, sends a *request* message to the temporal token holder. (tph enqueue it to token Q). However if AV_i already received a *declare* message or another *notify* message, it ignores the *notify* message received afterwards.

(3) generating a token
AV_i generates a *token* only when two conditions below are met at the same time;
·AV_i arrives at the CZ entry border.
·AV_i does not receive a declare message or a notify message before the timeout *tmt* occurs.
All AVs that satisfy the conditions above should broadcast a *location* message. If the number of AVs are more than two, the token generator will be decided as below;
AV_i that is located in the stream having the lowest number generates a *token* and declares the session.

3.2.2 Processing Section

Processing Section is the procedure in which AV_i in *waiting* state enters the CZ and passes through the IC. AV_i can enter the CZ only if it receives the *token* or listens to the session declare.

(1) *receiving the token*
AV_i, on receiving the *token*, starts the procedure of the CZ entering as follows:

①session declare
On receiving the *token*, AV_i declares the session $session_i$ by a *declare* message to all other AVs in the IC. We call AV_i declaring the session *key AV*.
The *declare* message is the form of (ID, PG) where ID is the AV's ID and PG is the passing group of the session.
On the other hand, if the tail of VPG_i receives the *token* temporarily, broadcasts its position by a *notify* message.

② *safety check*
AV_i , on receiving the *token* from the previous token holder AV_j , may not be able to use the token immediately. This is because AVs belong to the previous session may not be able to exit the CZ yet. Certainly, AV_i should check the situation of the CZ before entering the CZ and wait until those AVs belong to the previous session have been released out before it can use the token to enter the CZ.
In *TDGim*, to check the safety of the session start, we utilize a sequence number $session_i$ that represents the session to which the *token* belongs. The sequence number is incremented by one when the token holder transfers the *token* to the next holder. AVs, on releasing the CZ, piggyback the number of the session it belongs on the *release* message it sends. AV_i records the number of the *release* messages it has received containing the most recent session sequence number (given by variables $noOfRelease_i$ and $session_i$). Further a token contains the number of <*session-declare*> messages that were sent for the previous session (given by variable *oldNoOfDeclare* in the token). To decide whether the CZ for the new session is safe, AV_i evaluates the following condition:
($noOfRelease_i = holder_i.oldNoOfDeclare$)

(Veeraj Mittal. *2007, A priority-based distributed group mutual exclusion algorithm when group access ic non-uniform*)

AV_i, on completing the preparation procedure of the CZ entering, goes into the CZ immediately and switches to *passing* state from *waiting.*

(2) receiving a request message
When the *key AV* or the temporal token holder AV_i receives a *request* message from other AV AV_j, it updates the *tsReq[i]* in the token and enqueues the request of AV_j into the tok_i,*reqQ* attached to the *token*.

(3) receiving a declare message
If AV_i does not get the token, it may receive a *declare* message from the token holder during the waiting time. On receiving the message, it is able to perceive the VPG of the new session and the ID of the *key AV* and records those information received. Any AVs receive the message can enter the CS for the same session simultaneously.
Note that, only the AV knowing the type of the session can join the session(i.e. can enter the CS). Therefore, other AVs that does not receive the *declare* message can not join the session at any case and wait.

3.2.3 *Exit Section*

Exit Section is the procedure in which AV_i exits the CS after passing through the CS. It is a critical process to decide the new leader of the next session and the performance of the system can be increased by designing the token rotating effectively. There are two cases as below.
(1) *case 1* : AV_i holds the token**.**
AV_i, when exiting the CS, if there is a vehicle in VPG_i, transfers the token to the tail of VPG_i , and if not, it transfers the token to the tail of $VPG_{(i+4)}$. On completing the CZ passing, it will be in contact with QBC and receive the boundary information from the sensor. Then, it shall proceed the CZ exit procedure as follows.

① AV_i, as the leader of the current session $holder_i$, should select an IL in which an AV becomes the token holder of the next session. The IL for the new session is selected according to the criterion described below and we call the IL selected for the next session *NIL*.

·*Criterion for selecting NIL*
In *TDGim,* each request is enqueued to the queue *token-queue*$_i$ attached to the token, i.e. all the requests are centralized at the token. So we define various NIL selection schemes on the queue of the token.
We can define various selection schemes
in the priority $P(IL_i)$ for each IL_i as follows :
$NIL(IL_i) = a\ IL(r_i) + b\ IL_{i+1} + c\ IL_{SZ}(r_i) + d\ IL_{AG}(r_i)$,
where the following hold :

·*a,b,c* and *d* are constant parameters.
·Order : $IL(r_i)$ is the reserved order of arrival of r_i at each IL counted by *tsReq*$_i$.
·Traffic rule : IL_{i+1} is the reserved order in numerical order of IL.
· IL size : $IL_{SZ}(r_i)$ is the number of pending requests that are in the same IL as r_i's.
· IL age : $IL_{AG}(r_i)$ is given by the sum of the *ages* of all the requests in each IL. (*age* is the number of sessions that have been initiated since the request was enqueued to *token-queue*$_i$.)

Base on this framework, various selection schemes can be obtained by setting four parameters as follows:

· *a*=1, *b*=0, and *c*=*d*=0.
This scheme is the simplest scheme and is the first-come, first-serve scheme in which requests are granted in the order of the queue. It yields lower concurrency, but is a non-starving scheme.

· *a*=0, *b*=1, and *c*=*d*=0.
This scheme is the same as the Korean intersection control scheme in which the token is rotated in IL numerical order. It would yield lower concurrency.

·*a*=0, *b=0*, and *c*=*d*=1.
When the QZ of the intersection is heavily loaded, it is desirable that the next session be of the IL for which the number of pending requests in the token queue *token-queue*$_i$ is the maximum, However this method would lead to starvation of a request. To avoid starvation, every pending request in *token-queue*$_i$ is associated with an attribute called *age*.
The priority of a type of the new session is computed by simply adding two parts such as $IL_{SZ}(r_i)$ and $IL_{AG}(r_i)$. The next session that is initiated of IL excluding the IL it located for which the priority value is the maximum called NIL. Finally, the one of the AVs (which is the oldest in *tsReqIL*$_i$) in the NIL *NIL-head* is selected to become the next token holder.

②Then, *AV*$_i$ transfers the token to the *NIL-head* by sending a <*send-pt*> message.

③Afterwards, *AV*$_i$ sends a release message to the token holder all AVs in the intersection and exits the CZ.

(2) *case 2* : *AV*$_i$ does not hold the token.

AV$_i$ sends a *release* messages to all AVs in the intersection and exits the CZ.

IV. Performance evaluation

TDGim is an algorithm that applies the GME theory to the intersection traffic control system of autonomous vehicles. We analyze the performance in the respect of the measures of a distributed GME algorithm : *message complexity, synchronization delay, concurrency and system throughput.*

4.1 *Message complexity*

Message complexity is the member of messages needed for AV_i from when entering QZ to when leaving CZ. *Our algorithm, TDGim* exchanges eight kinds of messages and there would be various set of messages based on the situation. Surely the number of messages to enter CZ is the highest when an AV broadcasts <*declare*> message.

As shown in *table 1*, the message complexity of *TDGim* is better than others considering the specialty for dynamic traffic processing at the intersection.

4.2 *Synchronization Delay*

Synchronization delay is the time between all the vehicles in a session exiting the CZ and a vehicle in next session entering the CZ. In the case of *TDGim*, the *synchronization delay* can be measured as the delay time due to the PT transmission to the next PT holder selected and is divided into two factors. The first is the network transmission delay (Pt), and the second is the computer processing time delay (Ps). Let the total delay time be P, P = Pt + Ps. The *synchronization delay* of *TDGim* is evaluated as the maximum value of Pt because Ps is negligible time.

4.3 Concurrency

Concurrency is defined by the maximum number of vehicles that can enter the CZ simultaneously. Thus, a high degree of concurrency refers to more efficient utilization of resources and the maximum concurrency of GME is **n** when **n** is the total number of processes in the system.

AVs in *TDGim* are divided into 8 ILs and le there be an equal amount of AVs in each IL. There are **n**/8 vehicles per IL. So, the maximum concurrency of *TDGim* is **n**/4.

<table 1> performance

	mc	sd	cr
TDGim	*n+a*	*t*	*n/8*
Mamun	*n+2*	*2t*	*n*
Mittal	*2n-1*	*t*	*n*
Weigang	*3n*		*n/8*
Kakugawa	*5Q+1*	*3-4t*	*n*

*mc : message complexity, sd : synchronization delay, cr : concurrency

*n: number of processes, Q: quorum, t: maximum message delay
(Hirotsugu Kakugawa, Sayaka Kamei, Toshimitsu Masuzawa, 2008)

V. Conclusion

TDGim applies a token-based algorithm in distributed AIM to reduce message traffic and to improve traffic flow efficiency.

Also, *TDGim* is designed to increase the traffic flow in the dynamic situation where AVs can not stay in CZ after passing through CZ. For this purpose, a token is rotated by the systematic method. The traffic throughput of *TDGim* is increased by delivering the token to the last AV in its session.

Furthermore, it is possible to apply *TDGim* at any intersection with more lanes as well as at small intersections without traffic lights.

Though our study is the beginning stage of D_AIM, it is meaningful as an underlying technology necessary for the future AV progress. In *TDGim*, CZ which is the critical section of the intersection is designed as one unit. However, in order to improve the system of traffic control at the intersection, it is necessary to divide the CZ into several units and apply the AV's speed in the CZ.

1 References

[1] Weigang Wu, Jiebin Zhang, Aoxue Luo, and Jiannong Cao (2015). "A Distributed mutual exclusion algorithm for intersection traffic control"

[2] Fei Yan, Mahjoub DRIDI, Abdellah el Moudni (2013). "An autonomous vehicle sequencing problem at intersections"

[3] Md. Abdus Samad Kamal, Jun-ichi Imura, Tomohisa Hayakawa, Akira Ohata (2015) "A vehicle-Intersection Coordination Scheme for Smooth flows of traffic without using traffic lights"

[4] Li Li, Fei-Yue Wang (2006) "Cooperative driving blind crossing using inter-vehicle communication". [5]Xiangjun Qian, Jean Gregoire, Fabien Moutarde, Arnaud De La Fortelle Cañete, (2014). "coordination of autonomous and legacy vehicles at intersection"

[5] Joyoung Lee, Byungkyu Park(2012). "Development and Evaluation of Cooperative vehicle Intersection control algorithm under the connected vehicles environment"

[6] Ismail H. Zohdy, Hesham A. Rakha (2016). "Intersection management via vehicle connectivity: The intersection cooperative adaptive cruise control system concept"

[7] Quazi Ehsanul Kabir Mamun, Hidenori Nakazato (2006). "A new token based Protocol for Group mutual exclusion in distributed systems", [9]Neeraj Mittal, Prajwal K. Mohan (2007). "A priority-based distributed group mutual exclusion algorithm when group access is non-uniform"

[8] Hirotsugu Kakugama, Sayaka Kemei, Toshimitsu Masuzawa (2008). "A token based distributed group mutual exclusion with quorum"

[9] Priyamvadha Thambu, Jonny Wong (1995). "An efficient token-based mutual exclusion algorithm in a distributed system"

[10] Glenn Ricart, Ashok K. Agrawala (1981). "An optimal algorithm for mutual exclusion in computer networks"

[11] Yuh-Jzer Joung (2003). "Quorum-based algorithms for group mutual algorithm"

[12] Ranganath Atreya, Neeraj Mittal, Sathya Peri (2007). "A quorum-based group mutual exclusion algorithm for a distributed system with dynamic group set"

[13] Ranganath Atreya, Neeraj MittalM (2005). " A dynamic group mutual exclusion algorithm using surrogate-quorums"

[14] K. M. Chandy, J. Miara (1984). "The drinking philosophers problem"

[15] Mamotu Maekawa (1985), "A root N algorithm for mutual exclusion in decentralized systems"

[16] Kerry Raymond (1989), "A tree-based algorithm for distributed mutual exclusion"

[17] Abhishek Swaroop, Awadhesh Kumar Singh (2009), "Efficient group mutual exclusion protocols for message passing distributed computing systems",

[18] Lamport, L. (1978), "Time, clocks, and the ordering of events in a distributed system"

[19] Chang Y.I., Singha, M., Liu, M.T. (1991), "A dynamic token-based distributed mutual exclusion algorithm"

[20] Bertier, M., Arantes,L., Sens, P., (2006), "Distributed mutual exclusion algorithms for grid applications: a hierarchical approach"