Distributed Mutual Exclusion Algorithms for Intersection Traffic Control

Weigang Wu, *Member, IEEE*, Jiebin Zhang, Aoxue Luo, and Jiannong Cao, *Senior Member, IEEE*

Abstract—Traffic control at intersections is a key issue and hot research topic in intelligent transportation systems. Existing approaches, including traffic light scheduling and trajectory maneuver, are either inaccurate and inflexible or complicated and costly. More importantly, due to the dynamics of traffic, it is really difficult to obtain the optimal solution in a real-time way. Inspired by the emergence of vehicular ad hoc network, we propose a novel approach to traffic control at intersections. Via vehicle to vehicle or vehicle to infrastructure communications, vehicles can compete for the privilege of passing the intersection, i.e., traffic is controlled via coordination among vehicles. Such an approach is flexible and efficient. To realize the coordination among vehicles, we first model the problem as a new variant of the classic mutual exclusion problem, and then design algorithms to solve new problem. Both centralized and distributed algorithms are. We conduct extensive simulations to evaluate the performance of our proposed algorithms. The results show that, our approach is efficient and outperforms a reference algorithm based on optimal traffic light scheduling. Moreover, our approach does not rely on traffic light or intersection controller facilities, which makes it flexible and applicable to various kinds of intersections.

Index Terms—Mutual exclusion, intersection traffic control, intelligent transportation system, vehicular ad hoc network, distributed algorithm

1 Introduction

The ever-increasing traffic congestions, accompanied by unpredicted emergencies and accidents have motivated the development of intelligent transportation systems (ITS) [38], [44], [45], [46]. ITS has various applications, ranging from traffic surveillance, collision avoidance, to automatic transportation pricing. Among others, traffic control at intersections has been always a key issue in the research and development of ITS [10]. Based on the mechanisms used, existing approaches can be categorized into two classes.

Traffic light scheduling is the traditional approach, where vehicles proceed in a stop-and-go style according to the occurrence of green light. Recent efforts on traffic light control focus on adaptive and smart traffic light scheduling, mainly by making use of computational intelligence [41], including evolutionary computation algorithm [8], [23], fuzzy logic [16], [29], neural network [3], [34] and machine learning [30]. Adaptive approach based on real-time traffic volume information collected from sensor networks has also been studied [43].

However, due to the dynamics of traffic load, traffic control systems are large complex nonlinear stochastic systems, so determining the optimal time of green light is really hard even if not impossible [41], [42]. Moreover, the complexity of computational intelligence algorithms makes them usually not applicable to real-time traffic light control. More recently, advanced sensing and communication technologies [37] enable real-time traffic-response green light control [22], [43]. The traffic light is scheduled under a certain control strategy according to real-time traffic data and predefined logic rules.

The other approach, trajectory maneuver [1], [6], [26] is totally different from traffic light control. An intersection controller is deployed to optimally manipulate vehicles' trajectories based on nearby vehicles' conditions so as to avoid potential overlaps. The vehicles and controller communicate via wireless links.

With trajectory maneuver, the vehicles can move smoothly without stop and thus may improve the efficiency of the system. Different methods have been studied to calculate the optimal trajectories, including cell-based [13], [26], merging [31], fuzzy logic [27], scenario-driven [15], global adjusting and exception handling [22], etc. Similar to optimal traffic light control, trajectory maneuver is a hard problem due to the complexity of trajectory calculation. Moreover, the dependence on centralized controller makes the approach costly and prone to single point of failure. Please refer to Section S1.1 of the supplementary for detailed discussion on existing intersection control approaches, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/ TPDS.2013.2297097.

Inspired by recent advance vehicle technologies, we propose a novel approach based on distributed coordination among vehicles. Vehicular ad hoc network (VANET) [17] enables a vehicle to communicate with other vehicles (V2V)

- *W. Wu, and A. Luo are with the Department of Computer Science, Sun Yat-sen University, Guangzhou 510006, and SYSU-CMU Shunde International Joint Research Institute, Shunde, China. E-mail: wuweig@mail.sysu.edu.cn, 906464981@qq.com.*
- *J. Zhang is with the School of Software, Sun Yat-sen University, Guangzhou, Guangdong 510006, China. E-mail: 497608623@qq.com.*
- *J. Cao is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: csjcao@comp.polyu.edu.hk.*

Manuscript received 19 Sept. 2013; revised 22 Dec. 2013; accepted 22 Feb. 2013. Date of publication 23 Feb. 2014; date of current version 5 Dec. 2014. Recommended for acceptance by W. Jia.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.2297097

or infrastructures (V2I) via wireless communications. Then, vehicles can not only collect information about themselves and their environment, but they also exchange this information in real time with other vehicles. On the other hand, sensor and embedded technologies are making autonomous vehicle (AV) more and more feasible and practical. For example, Google's driverless car [48] has been running on the road.

With the help of above technologies, we propose to control the vehicles at intersections by letting vehicles compete for the privilege of passing via message exchange. A vehicle sends request to others and it can pass after permission from others is collected. How to realize such coordination of privilege control is at the core of our approach.

We model the problem as the vehicle mutual exclusion for intersections (VMEI) problem, a new variant of the mutual exclusion (MUTEX) problem [21], [25], [33]. In the classic MUTEX, all nodes access the critical section (CS) mutual exclusively, i.e., at most one node can be in CS at any moment. Although researchers have found and defined several other variants of MUTEX, including k-MUTEX [5], [12], [32], group MUTEX [4], [19], [20], the dining problem (DP) [9], [11], [24], [35], [36], the drinking problem [7], [14], [28], [39], and local MUTEX [2], they cannot describe the problem of traffic control at intersections. Please refer to Section S1.2 of the supplementary file for the detailed discussion on existing MUTEX variants, available online.

Then, to solve the VMEI problem, we design two algorithms for VMEI. The first one is a centralized algorithm, where a control center node is deployed at the intersection area. The control node communicates with vehicles and controls them to pass the intersection successfully. The other algorithm is a purely distributed algorithm, which does not rely on any centralized facility. The vehicles compete by exchanging messages among themselves and the order of passing is determined according to arrival time.

Two issues are addressed in our algorithm design. 1) Handling permission among dynamic changing set of vehicles, because each vehicle will leave after passing the intersection. 2) Achieving concurrency to improve the overall system efficiency.

Compared with existing intersection control approaches, including traffic light control and trajectory maneuver, our MUTEX based approach is quite different. It directly controls individual vehicles, which is similar to trajectory maneuver, but vehicles move in a stop-and-go style, as in traffic light system. Moreover, different from either of existing approaches, our distribute algorithm coordinate competition in an ad hoc way, without any centralized facility is involved.

Accordingly, our approach has two major advantages. 1) It is efficient and flexible, because the vehicles are directly controlled with real-time information and resources are fully used. 2) It is simple and not costly, because no optimization mechanism is involved and no centralized facility is necessary.

To evaluate the performance of our approach, we conduct extensive simulations. Besides our proposed algorithms, a traffic light based algorithm is also simulated as the reference. Several metrics are considered, including system throughput, vehicle waiting time, queue length, etc. The results show that our algorithms can handle various traffic volumes very well, and compared with the reference algorithm, they can reduce waiting time significantly.

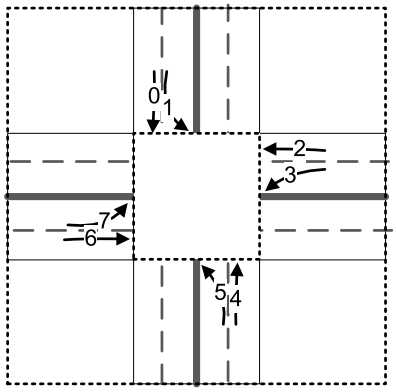


Fig. 1. The intersection.

In summary, our major contributions include:

1. The approach of controlling individual vehicles at intersection via wireless communications rather than traffic lights.
2. The definition of VMEI, a new MUTEX problem, which abstracts the problem of controlling vehicles to pass intersections.
3. Two algorithms to solve the VMEI problem: one is centralized and the other is distributed.
4. Extensive simulations to evaluate the performance of the proposed algorithms, with comparison with existing traffic light approaches.

The remainder of the paper is organized as follows. In Section 2, we describe the system model and preliminary definitions, and formally define the VMEI problem. The two proposed algorithms for VMEI are presented in Section 3 and Section 4 respectively. Section 5 presents performance evaluation via simulations. Finally, Section 6 concludes the paper and points out directions of future work.

2 System Model and Problem Definition

2.1 System Model and Definitions

2.1.1 The Intersection and Lanes

As shown in Fig. 1, we consider a typical intersection with four directions, i.e., north, south, east, and west. In each direction, there are two lanes, for going forward and turning left respectively. For the simplicity of presentation, the lanes are numbered from 0 to 7, and denoted by l_0 to l_7 respectively. Obviously, the path of a vehicle is determined by the lane it is at.

The small dashed rectangle represents the core area of the intersection. A vehicle in this area is called to be "passing" the intersection. The large dashed rectangle represents the queue area. A vehicle in this area is viewed as in the queue to pass the intersection.

Definition 1. *The concurrency/conflict relationship.*

According to road rules, vehicles with crossing paths have to pass the intersection mutual exclusively. Such

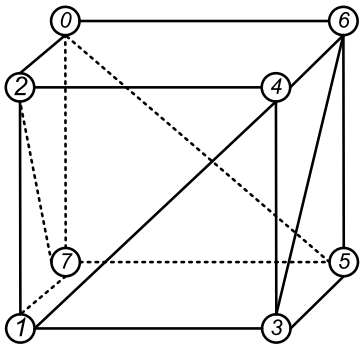


Fig. 2. The conflict graph.

vehicles are said to be "conflicting". Accordingly, the lanes of conflicting vehicles are also conflicting with each other.

On the other hand, vehicles with non-crossing paths can pass the intersection simultaneously. These vehicles and their lanes are to be "concurrent". Moreover, the vehicles at the same lane are viewed as concurrent with each other (they in fact proceed as a queue but here we ignore such order).

The concurrency and conflict relationship is denoted by "$\approx$" and "$\propto$" respectively. For example, we have $l_0 \approx l_4$, $l_0 \propto l_2$.

Definition 2. *The conflict graph of lanes, GL.*

The relationship among eight lanes can be clearly represented by a conflict graph GL, which is exactly a cube with one diagonal line at each side face, as shown in Fig. 2. The vertices represent the lanes at an intersection and the edges (dashed and real lines) represent the "conflict" between lanes. Two lanes without connecting edge are concurrent lanes.

Definition 3. *The strong concurrency relationship.*

Two lanes are strong concurrent with each other if they are represented by two vertices of the same face in the conflict graph and not directly connected by any edge. The strong concurrency relationship is denoted by "$\cong$". For example, we have $l_2 \cong l_3$.

2.1.2 The Vehicles and Communication Network

Each vehicle has a unique *id*, which can be the license plate number. The vehicles are AVs driven by autopilots.[1] Also, we assume there are sensors and other necessary devices for collision avoidance and navigation. A vehicle can get the knowledge of its lane number based on the digital map or other methods.

The vehicle is assumed to be able to detect the boundary of the queue/core area when it crosses the boundary. (This can be realized by deploying sensors at the boundary or making use of positioning system like GPS [37].)

Please notice that we do not assume a vehicle can recognize other vehicles queuing at the intersection by sensing because this is costly and not always feasible.

Wireless communication devices onboard enable vehicles to communicate with each other by sending and receiving messages. The *id* of the vehicle can be used as the address for communication. We assume that the transmission range of the communicating device is larger than the length of the queue area. That is, the vehicles inside the queue area constitute a one-hop ad hoc network and the each vehicle pair can communicate directly. We assume the wireless channel is FIFO channel.

1. In fact, our design is also applicable to vehicles driven by human. The only difference is that state switch commands need to be visible to the driver and she/he can take corresponding driving action.

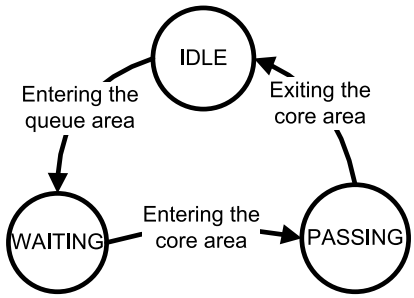


Fig. 3. The state transition diagram.

Please notice that, we do not consider message loss in our algorithm design, due to several reasons. 1) Although wireless network is prone to packet losses, one hop communication is much more reliable than general multi-hop networks. 2) As existing works of trajectory maneuver [22], the major contribution of this paper is the new approach of intersection control, and implementation issues like message loss are not the focus of our work. In fact, many mechanisms, such as timeout-based retransmissions, can be used to make the communication reliable. 3) Even if messages may still be lost with reliable mechanisms, collision avoidance mechanism equipped with AVs can be the last measurement to guarantee safety.

The procedure of a vehicle passing the intersection can be described by an automaton with three states, as shown in Fig. 3.

- IDLE: A vehicle is in the idle state if it is out of the queue area. That is, when a vehicle has not entered the queue area or it has passed the core area, it will be "idle".
- WAITING: The state that a vehicle waits for the permission to enter the core area.
- PASSING: The state that a vehicle is moving to pass the intersection. A vehicle is in the passing sate during the time interval between receiving the permission and exiting the core area.

2.2 The VMEI Problem

In the classical mutual exclusion (MUTEX) problem, at most on process can be in the critical section at any moment. All the processes compete with each other for the privilege to enter critical section. This is similar to the traffic control at intersection, where the vehicles cannot pass the intersection at the same time and they have to compete to obtain the privilege to pass the core area.

On the other hand, the intersection control problem is different from the MUTEX problems. First, the vehicles come and leave dynamically and pass only once. Second, vehicles at concurrent lanes can pass simultaneously. Third, vehicles at the same lane should pass as a group.

By considering the new requirements of traffic control, we define a new variant of MUTEX, i.e., the problem of vehicle mutual exclusion for intersections.

Definition 4. *The VMEI problem.*

Each vehicle at the intersection requests to pass the intersection along the direction as it wants. Accordingly, vehicles queue up at the corresponding lanes when they enter the queue area. To avoid collision or congestion, vehicles can pass the intersection simultaneously if and only if they are in concurrent lanes or the same lane.

The VMEI problem can be formally defined by the following correctness properties:

Safety (*mutual exclusion*): At any moment, if there is more than one vehicle in the core area, they must be concurrent with each other.

Liveness (*deadlock free*): If no vehicles are in the core area, some waiting vehicle must be able to enter the core area in a finite time.

Fairness (*starvation free*): Each vehicle must be able to pass the intersection after a finite number of vehicles do so.

Although MUTEX has several variants [2], [4], [32], they do not describe the complex situation at intersection, and new algorithms should be specially designed. Please refer to the related work section (Section S1) in the supplementary file for detailed discussion about this, available online.

3 The Centralized Algorithm

Inspired by traffic lights, we propose to solve the VMEI problem using a centralized facility. The lanes are controlled by locks and a vehicle can move only if it locks the lanes requested. To do so, we can deploy a centralized controller to manage the locks.

Vehicles request to lock the lanes by sending messages to the controller, and based on the response messages, they can get the privilege to pass. Since different vehicles request different lanes and vehicles may pass concurrently, how to guarantee mutual exclusion and at the same time achieve efficiency is at the core of the design.

In the following, we present the details of the centralized algorithm, including the design of locks and operations. Since the correctness of the centralized algorithm is obvious, we omit the proof to save space.

3.1 The Locks

There are totally eight locks for an intersection, each of which corresponds to a lane. The locks are maintained by the lock controller. It can be a network node deployed at the intersection.

The key point is to determine which lanes to be locked for a vehicle. A trivial solution is locking the vehicle's lane and all the four conflicting lanes shown in Fig. 1. However, with such design, vehicles at different lanes cannot pass the intersection even if they are in concurrent lanes. This will reduce the efficient of the whole system.

After examining all possible concurrent lane pairs, we propose to lock two conflicting lanes. The complete locking pattern is shown in Table 1. Then, a vehicle can pass the intersection if and only if the controller locks the corresponding lanes successfully.

TABLE 1
Locks Corresponding to Lanes

Lane of vehicle	Lanes to be locked	Lane of vehicle	Lanes to be locked
0	6, 7, 0	1	1, 2, 3
2	0, 1, 2	3	3, 4, 5
4	2, 3, 4	5	5, 6, 7
6	4, 5, 6	7	7, 0, 1

3.2 Notations and Message Types

The following notations are used in our design:

- st_i: the current state of vehicle i.
- lid_i: the lane number of vehicle i.
- lsl_i: the set of lanes to be locked for vehicle i, as shown in Table 1.
- *rp*: the list of pending requests, maintained by the controller.

The following message types may be used in our centralized algorithm:

- *REQUEST(vid, lid)*: The message sent from a vehicle to the controller to lock the lanes requested. REQUEST is sent when a vehicle enters the queue area. The parameter *lid* is the lane number of the sender and *vid* is the id of the vehicle.
- *PERMIT(plt)*: The message sent from the controller to grant the privilege of passing. This is sent when the lanes requested by some vehicle are locked successfully. A vehicle list *plt* is carried by this message, which contains the vehicles that can pass together under this permit.
- *RELEASE(vid, lid)*: The message sent from a vehicle to unlock the lanes locked. It is sent when a vehicle exits the core area.

Algorithm 1.1 The Centralized VMEI Alg. - Vehicle

```
CoBegin //for each vehicle i
On entering the monitoring area:
    send REQUEST(i, lid_i);
    st_i =WAITING;
    wait for PERMIT from Controller;
On Receiving PERMIT(plt)
    if ( i ∈ plt ){
        st_i =PASSING;
        move and pass the core area;
    }
On exiting the intersection
    if(i is the last vehicle in pass list)
        send RELEASE(i, lid_i) to the Controller;
    st_i=IDLE;
CoEnd
```

3.3 Algorithm Operations

The centralized algorithm consists of two parts: the vehicle part (Algorithm 1.1) and the controller part (Algorithm 1.2).

3.3.1 Operations of Request

When a vehicle i enters the queue area, it switches from IDLE to WAITING and sends REQUEST(i, *lid*) to the controller.

Then, when the controller receives the request message, it will try to lock the corresponding locks (lanes) according to Table 1. If some lane requested has been locked by vehicles at other lanes, the locking fails and the request will be queued in the pending request list.

If the locking succeeds, the controller will broadcast a PERMIT message to all the vehicles, with a list of vehicles that can pass under this permit. The number of vehicles in the list is bounded by a threshold value *np*, which can be determined by user or the traffic load. The controller records the *id* of the last vehicle in the permission list for later checking of unlocking.

Algorithm 1.2 The Centralized VMEI Alg. - Controller

```
On Receiving REQUEST(i, lid_i) from vehicle i
    if (all the lanes in lsl_i are locked by vehicles at the
                                    same lane and |plt| <np){
        add i to plt;
        broadcast PERMIT(plt);
    }
    if ( all the lanes in lsl_i are unlocked ){
        lock the lanes in lsl_i;
        construct the pass list plt by adding vehicles in the
                                    same lane (at most np vehicles);
        broadcast PERMIT(plt);
    } else
        add i to the pending list rp;
On Receiving RELEASE(i, lid_i)
    if (i is the vehicle recorded for unlocking lanes in lsl_i){
        unlock lanes in lsl_i;
        for each vehicle j that j ∈ rp do
            if (lanes in lsl_j are all unlocked){
                lock lanes in lsl_j;
                construct the pass list plt by adding
                        at most np vehicles in the same lane;
                broadcast PERMIT(plt);
            }
```

If the locks requested has been locked by vehicles at the same lane and the length of *plt* in previous PERMIT is less than *np*, the locking also succeeds. That is, the request will be met by sending a PERMIT message.

3.3.2 Operations of Passing

On receiving PERMIT (*plt*) from the controller, vehicles included in *plt* start to pass the intersection.

During such moving, the driver or the autopilot is in charge of handling collision avoidance and navigation.

When the last vehicle in *plt* reaches the exit point of the intersection, it sends a RELEASE (*vid*, *lid*) message to the controller.

3.3.3 Operations of Release

Upon receiving the RELEASE message, the controller will check if the sender is the vehicle recorded for unlocking. If it is not the one, the message is ignored.

Otherwise, the controller unlocks the corresponding locks and then checks if any request in pending list can be satisfied. It will lock the lanes for that request, construct pass list and broadcast the PERMIT(*plt*) message.

If more than one pending request can be met, the controller will handle them one by one, by traversing the pending list. Obviously, the early request will be met first. Such operation is necessary to allow concurrency passing and achieve high efficiency.

4 The Distributed Algorithm

Although the centralized algorithm is quite simple in terms of operations, it also has some problems. First, deploying a centralized control node is costly and not always feasible, because it requires additional device and installation work. Second, the centralized design is not flexible. In the real world, the road may have various shapes of intersections, so the controller and the locks must be delicately designed and configured for different intersections. Once the controller is deployed, it is not easy to change or upgrade it. Last, but not the least, the failure of the controller will cause serious problem, even if the many fault tolerance techniques can be used to enhance its reliability.

To avoid these problems, we have designed a distributed algorithm, where the vehicles coordinate by only exchanging messages among themselves. The major difficulty lies in the dynamics of vehicles. The vehicles arrive and leave from time to time. No vehicles will be always there. From which vehicles to obtain the privilege of passing must be carefully considered. An even more difficult issue is how to handle various cases of concurrent vehicles, without affecting the liveness property.

The basic idea of our design is as follows. Vehicles have different priorities to pass, which is generally determined by the arrival time. A vehicle broadcasts request message and the receivers with higher priority will prevent the sender via response message. If no receivers prevent the sender, it can pass the intersection. We also design mechanism to achieve high efficiency by allowing vehicles at concurrent lanes pass simultaneously. The pseudo code of the distributed algorithm is listed as Algorithm 2. The correctness proof is presented in Section S2 of the supplementary file, available online.

4.1 Notations and Message Types

Same as in the centralized design, a vehicle undergoes three phases to pass the intersection and correspondingly, there are three states: IDLE, WAITING and PASSING.

Each vehicle i maintains the following notations or data structures to execute the distributed algorithm:

- HL_i: High list, the list of vehicles that have a higher priority than i to pass the intersection. The priority is generally, but not always, determined by the arrival time of vehicles.
- LL_i: Low list, the list of vehicles that have a lower priority than i to pass the intersection.
- *CntPmp*: This is a counter to record the number of preemptions occurred at i. A preemption means a vehicle allow another with low priority to pass first at some special situation.

Algorithm 2. The Distributed VMEI Alg.

CoBegin //for a vehicle *i*
On entering the monitoring area:
st_i =WAITING;
broadcast REQUEST(*i*, lid_i);
wait for REJECT from others;
On Receiving REQUEST(*j*, *lid*) from *j*
if((st_i = WAITING|PASSING) and
$(lid_i = lid_j \vee lid_i \propto lid_j)$){
if($(\exists k, k \in HL_i \wedge j \cong k)$ and *CntPmp*<*TH*){
add *j* to HL_i;
CntPmp++;
} **else** {
add *j* to LL_i;
broadcast REJECT(*i*, *j*);
}
}
On Receiving REJECT(*j*, *k*) from *j*
if(st_i=WAITING){
if (*i*=*k*) add *j* to HL_i;
if(*i*≠*k* and ($k \in HL_i$ with preemption) and
$((j \propto i \wedge j \notin HL_i) \vee j \cong i)$){
delete *k* from HL_i;
broadcast REJECT(*i*, *k*);
}
}
On Receiving PERMIT(*j*) or timeout *tmt* occurs (no JECECT received)
delete *j* from HL_i;
if(HL_i is empty){
st_i=PASSING;
construct the follow list *flt*;
($flt = \{v \mid lid_v = lid_i \wedge v \in LL_i \wedge flt\text{'s length} < NF\}$)
broadcast FOLLOW(*i*, *flt*);
}
move and pass the core area;
On Receiving FOLLOW(*j*, *flt*) from *j*
if($i \in flt$){
st_i =PASSING;
move and pass the core area;
}**else if (** $i \propto j$ **)** {
delete *j* from HL_i;
delete vehicles in *flt* from HL_i or LL_i;
add the last one in *flt* to HL_i;
}
On exiting the intersection
if(the passing is triggered by a FOLLOW(*x*, *flt*) and *i* is the last in *flt*)
broadcast PERMIT(*i*);
CoEnd

The message types involved in the distributed algorithm are described as follows:

- *REQUEST (vid, lid)*: The message broadcasted by a vehicle to other vehicles to get the privilege of passing. REQUEST is sent when a vehicle enters the queue area. The parameters *vid* and *lid* are vehicle id and lane number respectively.
- *REJECT* (i, j): The message sent by vehicle i to block vehicle j when i has a higher priority than j. It is a response to a REQUEST message.
- *PERMIT* (i): This message sent by vehicle i to unblock the vehicles blocked by i.
- *FOLLOW* (i, flt): The message to let vehicles included in *flt* pass together with i. Namely, they can pass the intersection *continuously* (without waiting for other vehicle to pass in the between). We call this procedure a "*pass*". The parameter *flt* is the list of vehicles that can follow i.

4.2 Algorithm Operations

4.2.1 Operations of Request

When vehicle i enters the queue area, it switches from IDLE to WAITING and broadcasts a request message REQUEST (i, lid_i). Then, i waits for REJECT messages from others after setting a time out *tmt*.

When a vehicle j, in the WAITING or PASSING state, receives the request message from i:

i. if i and j are in concurrent lanes, j will ignore the message;
ii. if i and j are in the same or conflicting lanes, j puts i in its low list and sends REJECT(j, i) to j. (if j is in PASSING and it has switched to PASSING by a FOLLOW(x, flt) message and j is not the last in *flt*, j will not send the REJECT message.)

On receiving the REJECT (j, i) message from j, i puts j in its high list HL_i.

Mechanism to enable concurrency. With the operations above, two vehicles in concurrent lanes may not be able to pass concurrently due to vehicles in conflicting lanes and with priority in the between. For example, when a vehicle u in *l0* is passing, vehicle v in *l1* may be blocked by w in *l7* if w is between u and v according to the priority (arrival time). This will reduce the efficiency of the whole system. To avoid such problem, we need additional operations (the statements under "receiving request" within frames in Algorithm 2). Then, the operation ii) changes to be:

iii. if i and j are in conflicting lanes but some vehicle k in j's high list is strong concurrent with i, j will give way to i, i.e., it puts i in its high list and will not send REJECT; otherwise, j puts i in its low list and sends RE JECT(j, i) to i.

The new operation allows concurrent passing by letting vehicles with priority in the between of concurrent ones give way. That is, preemptions are allowed. On the other hand, such preemptions may cause starvation if a vehicle is always be preempted by vehicles at two concurrent lanes alternatively. To address this problem, a counter *CntPmp* is used to record how many times a vehicle has been preempted. If the value reaches the threshold *TH*, the vehicle will not give way anymore. *TH* can be static or adaptive to the real-time traffic flow.

Moreover, the preemption may also cause deadlock if two strong concurrent or conflict vehicles take different actions on the preemption of another vehicle. To avoid such problem, we design operations to coordinate the grant of preemption (the statements under "receiving REJECT" within frames in Algorithm 2):

When some REJECT(j, k) is received and i has put k to its high list due to preemption, i will cancel the preemption of k, if a) j is a conflicting vehicle and not in HL_i, b) j is strongly concurrent with i.

4.2.2 Operations of Passing

If no REJECT messages are received before the timeout *tmt* occurs, or the high list becomes empty upon receiving PERMIT messages, i switches to be PASSING and starts to pass the core area. If there are other waiting vehicles in the same lane (they must be in low list), i will add these vehicle to the follow list *flt* and broadcasts FOLLOW$(i,$ *flt*$)$.

To avoid starvation, the length of *flt* should be bounded by a threshold *np*, which is the same as in centralized algorithm. The value of *np* can be a static value determined based on the historic data, or a value adaptive to the real time traffic volume.

On receiving a FOLLOW(i, *flt*) message at j, if j is included in *flt*, j will empty its high list, switch to PASSING and starts to pass the core area of the intersection. Please notice that, in this case, j will not further construct and send FOLLOW message. Such a design is to avoid starvation at other lanes.

If the receiver of FOLLOW (i, *flt*), say w, is conflicting with i, w deletes i and vehicles in *flt* from its high/low list, and then puts the last vehicle in *flt* into its high list.

4.2.3 Operations of Release

After i passes the core area (crosses the exit boundary), i will empty its low list. It will also broadcast a PERMIT (i) message UNLESS its passing has been triggered by FOLLOW $(x,$ *flt*$)$ and i is not the last in *flt*.

On receiving the PERMIT (i) message at j, if i is in the high list of j, j deletes i from its high list. If the high list becomes empty, it will switch to PASSING.

5 Performance Evaluation

We examine the performance of our proposed algorithms by simulations. Since our approach is based on communications among vehicles, we conduct simulations using the popular network simulator ns-3 [47]. We did not use traffic simulators such as AIMSUN or VISSIM because they do not simulate communication among vehicles, which is the basis of our design. That is, we need a network simulator more than a traffic simulator.

To compare with existing approach, we also simulate an adaptive traffic light control algorithm as proposed in [43]. We do not choose computational algorithms due to their high complexity and infeasibility to real-time control. The traffic light is simulated as a centralized facility node in the network. The green light is scheduled adaptively according to the real-time traffic volume at each lane. A vehicle will pass the intersection if the green light for its lane is on and it is in the head of the queue. For simplicity of simulation, we omitted the sensors and the traffic volume information is set to be visible to the traffic light node.

TABLE 2
Key Parameters in Simulations

Parameters	Values
Territory of Intersection	100m*100m
Transmission range	200m
Communication protocol (MAC)	IEEE 802.11 (Ad hoc mode)
Time of passing core area	3s (go straight) 4s (turn left)
Traffic volume	8/min ~ 64/min
# of vehicles allowed in one pass, *np*	3
Simulation time	1200s
Timeout for REJECT	2s

5.1 Setup and Metrics

The major parameters are listed in Table 2. We simulate an intersection of 100 * 100 m. The transmission range of the communication device is set to be 200 m. We adopt IEEE 802.11 as the communication protocol. Since the network is one-hop, no routing protocol is necessary. All the packets are sent using broadcasting mode in MAC layer. That is, all the vehicles in the network can receive the packet even if they are not the destined node.

Since our simulation focus on the communication among vehicles, we set the network nodes (vehicles) to be static. The movement is simulated by controlling the duration of different states.

Traffic volume is set to be static for each run. Same as in most intersection control works, we vary the volume-to-capacity ratio (v/c) of each road to examine performance under different traffic load levels, and the capacity of the intersection is set to be 64 vehicles/min. The v/c value in our simulations is changed from 0.125 to 1.0 (i.e., 8 to 64 vehicles per minutes), a full scope of traffic load.

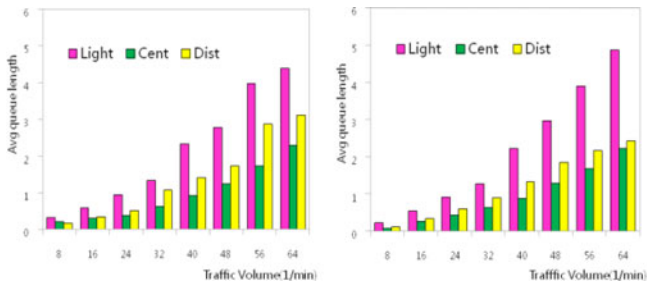
To examine the effect of distribution of traffic volume, we set two patterns. One is uniform volume, i.e., each lane has the same vehicle arrival rate. The other one is non-uniform volume, where the volume in south-north road is three times of that in west-east road.

To measure the performance of the traffic control algorithms, the following metrics are used.

- *Average queue length*: the average length of the queue of waiting vehicles in each lane.
- *Average waiting time*: the average time duration that a vehicle stays in WAITING state.
- *System throughput*: the total number of vehicles that pass the intersection in per minute.
- *Message cost*: the number of messages sent for passing the intersection once per vehicle.

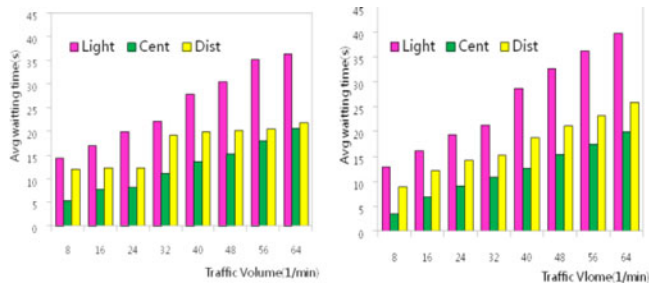
5.2 Results and Analysis

We present and analyze simulations results according to performance metrics one by one. The results from our



(a) Uniform volume (b) Non-uniform volume

Fig. 4. Average queue length.



(a) Uniform volume (b) Non-uniform volume

Fig. 5. Average waiting time.

centralized algorithm, distributed algorithm and the traffic light based algorithm are labeled "Cent", "Dist" and "Light", respectively.

5.2.1 Average Queue Length

Results of average queue length are shown in Fig. 4. With the increase of traffic volume, the queue length increase accordingly. The reason is obvious. With a higher volume, more vehicles want to pass the intersection and more vehicles are in waiting state.

Among the three algorithms, our centralized algorithm performs the best and the traffic light based algorithm is the worst. The difference among the algorithms increases with the increase of traffic volume. This obviously shows that our MUTEX based approach is more efficient than traditional approach of traffic light. The centralized algorithm outperforms the distributed one due to the control node has global information and can make better decision on permitting which vehicle to pass.

Comparing Figs. 4a and 4b, we can see the effect of traffic volume pattern, i.e., the distribution of traffic volume among different lanes. It is interesting to see that the traffic light approach has a longer queue under non-uniform traffic volume pattern; while our proposed algorithms achieve shorter queue length under non-uniform traffic volume. This can be explained as follows. In our approach, with non-uniform traffic volume, the vehicles in the busy direction can pass concurrently with larger chance. However, the traffic light approach schedule green light patterns one by one without considering such dynamics of concurrency of vehicles.

5.2.2 Average Waiting Time

Fig. 5 shows the results of average waiting time. Same as the change of queue length, waiting time also increases as the traffic volume increases. Generally, queue length dominates the value of waiting time.

With our proposed approach, the waiting time is between 5 and 25 s. Such a value is reasonable considering the real world's transportation. With non-uniform traffic volume, the waiting time is shorter that uniform case. As aforementioned, with non-uniform volume, the vehicles in the busy direction can pass concurrently with larger chance.

Compared with the traffic light algorithm, both our algorithm achieves shorter waiting time. Our distributed algorithm gets longer waiting time than the centralized one. This is reasonable because the centralized controller maintains the global view of the intersection and can grants passing with higher concurrency.

5.2.3 System Throughput

Results of system throughput are shown in Fig. 6. With the increase of traffic volume, the throughput also increases. Obviously, the algorithms can handle the traffic very well and the system is not saturated, even if the traffic volume reaches 64 vehicles per minute, which is a quite high volume considering the real world traffic situation.

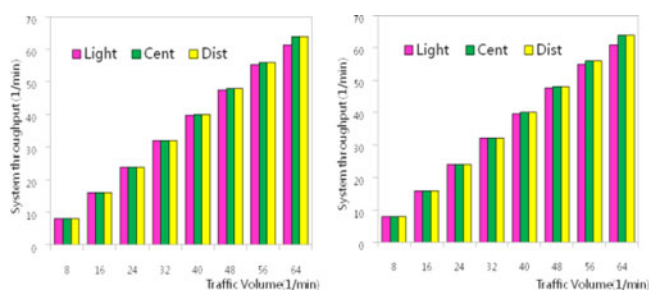
Compared with our algorithms, the traffic light algorithm achieves nearly the same throughput under low traffic volume. When the volume is high (>32/min), the throughput of our approach achieves higher throughput than the traffic light approach.

However, the advantage of our approach in throughput is not as large as that in waiting time. This is because, when the intersection is not saturated, a shorter waiting time may not result in a higher throughput. For example, two different scheduling approaches both let 10 vehicles (arriving at the same moments) pass within one minute, but they may schedule them to pass at different moments. Then, the two algorithms achieve the same throughput but different waiting time values.

5.2.4 Message Cost

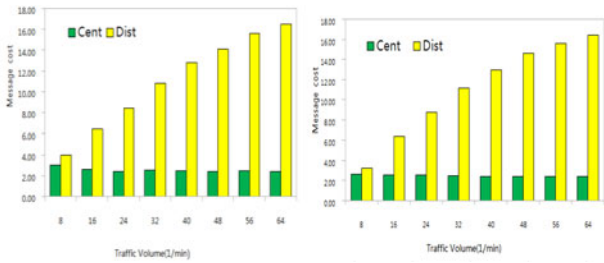
Fig. 7 shows the results of message cost. Since the traffic light approach relies on traffic signal rather than message passing, this metric is for only our two proposed algorithms.

Roughly, with the centralized algorithm, a vehicle needs to send about three messages per passing. On the other hand, the distributed algorithm costs more messages, varying from 4 to 16 messages. Such message costs are



(a) Uniform volume (b) Non-uniform volume

Fig. 6. System throughput.



(a) Uniform volume (b) Non-uniform volume

Fig. 7. Message cost.

reasonable and the difference is expected. In the centralized algorithm, one vehicle needs communicate with only the central node and at most three messages (one REQUEST, one PERMIT and one RELEASE) are sent for each pass of a vehicle. However, the distributed algorithm needs to send more messages, especially REJECT messages, due to the lacked of a centralized control node.

It is interesting to examine the effect of traffic volume. With the traffic volume increases, the message cost of the distributed algorithm increases while the value of the centralized algorithm decreases. For the distributed algorithm, more traffic results in more conflicts and consequently more messages. However, for the centralized algorithms, large volume increases the possibility of following vehicles, so fewer messages are sent by the controller.

6 Conclusion and Future Work

In this paper, we propose a novel approach to the problem of traffic control at intersections. Rather than solving the optimization problem of green light scheduling or trajectory maneuver, we let the vehicles compete for the privilege of passing by message exchange. We first model the intersection control problem as a new variant of the classic mutual exclusion problem in distributed computing, with new properties. Then, we design algorithms to solve the new problem and formally prove their correctness. Performance evaluation by simulations shows that our algorithms can handle various traffic cases very well, with very little message cost. Compared with the adaptive traffic light based approach, which is widely studied, our approach can achieve much higher efficiency.

Our approach is easy and simple since no optimization problem is involved. Moreover, it does not rely on deployment of traffic light facilities or intersection controller, which makes it flexible and applicable to various kinds of intersections.

Since this is the initiative work of a new intersection control approach, further efforts are obviously necessary to extend and improve it. Interesting issues include reducing message cost, handling networked intersections, handling exceptional actions, and extending the VMEI problem to a general version for different applications, etc.

Acknowledgments

This research was partially supported by National Natural Science Foundation of China (No. 61379157), Guangdong Natural Science Foundation (No. S2012010010670), Pearl River Nova Program of Guangzhou (No. 2011J2200088), and National High-Tech Research and Development Program ("863") of China (No. 2013AA01A212).

References

[1] B. van Arem, C.J.G. van Driel, and R. Visser, "The Impact of Cooperative Adaptive Cruise Control on Traffic-flow Characteristics," *IEEE Trans. Intelligent Transportation Systems*, vol. 7, no. 4, pp. 429-436, Dec. 2006.

[2] H. Attiya, A. Kogan, and J.L. Welch, "Efficient and Robust Local Mutual Exclusion in Mobile Ad Hoc Networks," *IEEE Trans. Mobile Computing*, vol. 9, no. 3, pp. 361-375, Mar. 2010.

[3] E. Bingham, "Reinforcement Learning in Neurofuzzy Traffic Signal Control," *European J. Operational Research*, vol. 131, pp. 232-241, 2001.

[4] V. Bhatt and C.-C. Huang, "Group Mutual Exclusion in O(log n) RMR," *Proc. 29th ACM SIGACT-SIGOPS Symp. Principles of Distributed Computing (PODC '10)*, pp. 45-54, July 2010.

[5] S. Bulgannawar and N.H. Vaidya, "A Distributed K-Mutual Exclusion Algorithm," *Proc. 15th Int'l Conf. Distributed Computing Systems (ICDCS '95)*, 1995.

[6] R.J. Caudill and J.N. Youngblood, "Intersection Merge Control in Automated Transportation Systems," *Transportation Research*, vol. 10, no. 1, pp. 17-24, Feb. 1976.

[7] K.M. Chandy and J. Misra, "The Drinking Philosopher's Problem," *ACM Trans. Programming Languages and Systems*, vol. 6, no. 4, pp. 632-646, 1984.

[8] X. Chen and Z. Shi, "Real-Coded Genetic Algorithm for Signal Timings Optimization of a Signal Intersection," *Proc. First Int'l Conf. Machine Learning and Cybernetics*, 2002.

[9] M. Choy and A.K. Singh, "Efficient Fault-Tolerant Algorithms for Distributed Resource Allocation," *ACM Trans. Programming Languages and Systems*, vol. 17, pp. 535-559, 1995.

[10] I. Day, "Scoot-Split, Cycle and Offset Optimization Technique," TRB Committee AHB25 Adaptive Traffic Control, 1998.

[11] E.W. Dijkstra, "Hierarchical Ordering of Sequential Processes," *Acta Informatica*, vol. 1, pp. 115-138, 1971.

[12] D. Dolev, E. Gafni, and N. Shavit, "Toward a Non-Atomic Era: l-Exclusion as a Test Case," *Proc. 20th Ann. ACM Symp. Theory of Computing (STOC '88)*, pp. 78-92, 1988.

[13] K. Dresner and P. Stone, "A Multiagent Approach to Autonomous Intersection Management," *J. Artificial Intelligence Research*, vol. 31, no. 1, pp. 591-656, Jan. 2008.

[14] D. Ginat, A.U. Shankar, and A.K. Agrawala, "An Efficient Solution to the Drinking Philosophers Problem and its Extensions," *Proc. Int'l Workshop Distributed Algorithms (WDAG '89)*, pp. 83-93, 1989.

[15] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouvelière, "Maneuver-Based Trajectory Planning for Highly Autonomous Vehicles on Real Road with Traffic and Driver Interaction," *IEEE Trans. Intelligent Transportation Systems*, vol. 11, no. 3, pp. 589-606, Sept. 2010.

[16] B.P. Gokulan and D. Srinivasan, "Distributed Geometric Fuzzy Multiagent Urban Traffic Signal Control," *IEEE Trans. Intelligent Transportation Systems*, vol. 11, no. 3, pp. 714-727, Sept. 2010.

[17] H. Hartenstein and K.P. Laberteaux, "A Tutorial Survey on Vehicular Ad Hoc Networks," *IEEE Comm. Magazine*, vol. 46, no. 6, pp. 164-171, June 2008.

[18] A.D. Joseph et al., "Intelligent Transportation Systems," *IEEE Pervasive Comp.*, vol. 5, no. 4, pp. 63-67, Dec. 2006.

[19] Y.-J. Joung, "Asynchronous Group Mutual Exclusion," *Proc. 17th Ann. ACM Symp. Principles of Distributed Computing (PDOC)*, June 1998.

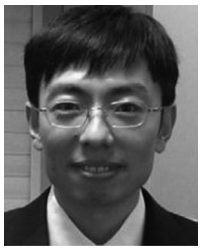
[20] P. Keane and M. Moir, "A Simple Local-Spin Group Mutual Exclusion Algorithm," *Proc. Ann. ACM Symp. Principles of Distributed Computing (PDOC)*, 1999.

[21] L. Lamport, "A Fast Mutual Exclusion Algorithm," *ACM Trans. Computer Systems*, vol. 5, no. 1, pp. 1-11, 1987.

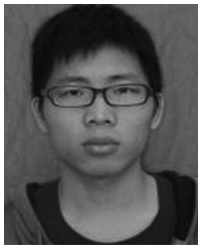
[22] J. Lee and B. Park, "Development and Evaluation of a Cooperative Vehicle Intersection Control Algorithm Under the Connected Vehicles Environment," *IEEE Trans. Intelligent Transportation Systems*, vol. 13, no. 1, pp. 81-90, Mar. 2012.

[23] P. Lertworawanich, M. Kuwahara, and M. Miska, "A New Multi-objective Signal Optimization for Oversaturated Networks," *IEEE Trans. Intelligent Transportation Systems*, vol. 12, no. 4, pp. 967-976, Nov. 2011.

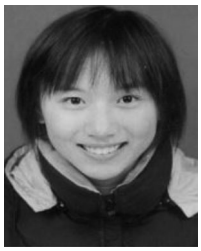
[24] N.A. Lynch, "Upper Bounds for Static Resource Allocation in a Distributed System," *J. Computer and System Science*, vol. 23, no. 2, pp. 254-278, 1981.
[25] M. Maekawa, "A $\sqrt{n}$ Algorithm for Mutual Exclusion in Decentralized Systems," *ACM Trans. Computer Systems*, vol. 3, no. 2, pp. 145-59, 1985.
[26] F.J. McGinley, "An Intersection Control Strategy for a Short-Headway P.R.T. Network," *Transportation Planning Technology*, vol. 3, no. 1, pp. 45-53, 1975.
[27] V. Milanés, J. Perez, E. Onieva, and C. Gonzalez, "Controller for Urban Intersections Based on Wireless Communications and Fuzzy Logic," *IEEE Trans. Intelligent Transportation Systems*, vol. 11, no. 1, pp. 243-248, Mar. 2010.
[28] S.L. Murphy and A.U. Shankar, "A Note on the Drinking Philosophers Problem," *ACM Trans. Programming Languages and Systems*, vol. 10, no. 1, pp. 178-188, 1988.
[29] L.A. Prashanth and S. Bhatnagar, "Reinforcement Learning with Function Approximation for Traffic Signal Control," *IEEE Trans. Intelligent Transportation Systems*, vol. 12, no. 2, pp. 412-421, June 2011.
[30] J. Qiao, N.D. Yang, and J. Gao, "Two-Stage Fuzzy Logic Controller for Signalized Intersection," *IEEE Trans. Systems, Man and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 1, pp. 178-184, Jan. 2011.
[31] G. Raravi, V. Shingde, K. Ramamritham, and J. Bharadia, "Merge Algorithms for Intelligent Vehicles," *Proc. GM R&D Workshop*, pp. 51-65, 2007.
[32] V.A. Reddy, P. Mittal, and I. Gupta, "Fair K Mutual Exclusion Algorithm for Peer to Peer Systems," *Proc. 28th Int'l Conf. Distributed Computing Systems (ICDCS)*, 2008.
[33] G. Ricart and A.K. Agrawala, "An Optimal Algorithm for Mutual Exclusion in Computer Networks," *Comm. ACM*, vol. 24, pp. 9-17, 1981.
[34] D. Srinivasan, M.C. Choy, and R.L. Cheu, "Neural Networks for Real-Time Traffic Control System," *IEEE Trans. Intelligent Transportation Systems*, vol. 7, no. 3, pp. 261-272, Sept. 2006.
[35] P.A. Sivilotti, S.M. Pike, and N. Sridhar, "A New Distributed Resource-Allocation Algorithm with Optimal Failure Locality," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS)*, pp. 524-529, 2000.
[36] Y.-K. Tsay and R. Bagrodia, "An Algorithm with Optimal Failure Locality for the Dining Philosophers Problem," *Proc. Eighth Int'l Workshop Distributed Algorithms (WDAG '94)*, pp. 296-310, 1994.
[37] M. Tubaishat, P. Zhuang, Q. Qi, and Y. Shang, "Wireless Sensor Networks in Intelligent Transportation Systems," *Wireless Comm. and Mobile Computing*, vol. 9, no. 3, pp. 287-302, Mar. 2009.
[38] F. Wang, "Parallel Control and Management for Intelligent Transportation Systems: Concepts, Architectures, and Applications," *IEEE Trans. Intelligent Transportation Systems*, vol. 11, no. 3, pp. 630-638, Sept. 2010.
[39] J.L. Welch and N.A. Lynch, "A Modular Drinking Philosophers Algorithm," *Distributed Computing*, vol. 6, no. 4, pp. 233-244l, 1993.
[40] W. Wu, J. Cao, and J. Yang, "A fault Tolerant Mutual Exclusion Algorithm for Mobile Ad Hoc Networks," *Pervasive and Mobile Computing*, vol. 4, no. 1, pp. 139-160, Feb. 2008.
[41] D. Zhao, Y. Dai, and Z. Zhang, "Computational Intelligence in Urban Traffic Signal Control: A Survey," *IEEE Trans. Systems, Man, and Cybernetics-Part C: Applications and Reviews*, vol. 42, no. 4, pp. 485-494, July 2012.
[42] L. Zhao, X. Peng, L. Li, and Z. Li, "A Fast Signal Timing Algorithm for Individual Oversaturated Intersections," *IEEE Trans. Intelligent Transportation Systems*, vol. 12, no. 1, pp. 280-283, Mar. 2011.
[43] B. Zhou, J. Cao, X. Zeng, and H. Wu, "Adaptive Traffic Light Control in Wireless Sensor Network-Based Intelligent Transportation System," *Proc. IEEE Vehicular Technology Conf. Fall*, Sept. 2010.
[44] SCATS, http://www.scats.com.au/, Nov. 2013.
[45] VICS, http://www.vics.or.jp/, Nov. 2013.
[46] VII, http://www.vehicle-infrastructure.org/, Nov. 2013.
[47] ns3, http://www.nsnam.org/, Nov. 2013.
[48] Google driverless car, http://www.wikipedia.org/, Nov. 2013.



Weigang Wu received the BSc and the MSc degree from Xi'an Jiaotong University, China, in 1998 in 2003, respectively. He received the PhD degree in computer science from Hong Kong Polytechnic University in 2007. He is currently an associate professor at the department of computer science, Sun Yat-sen University, China. His research interests include distributed systems and wireless networks, especially cloud computing platforms and ad hoc networks. He has published more than 50 papers in conferences and journals. He has served as a member of editorial board of two international journals, *Frontiers of Computer Science*, and *Ad Hoc & Sensor Wireless Networks*. He is also an organizing/program committee member for many international conferences. He is a member of the IEEE and ACM.



Jiebin Zhang received the BSc degree in computer science from the Guangdong University of Technology, and MSc degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2011 and 2013, respectively. He is currently a programmer at Guangzhou Rural Commerce Bank, China. His research interests include distributed algorithms and their applications.



Aoxue Luo received the BSc degree in computer science, from Northeast Normal University, Changchun, China, in 2011. She is currently working toward the MSc degree in Sun Yat-sen University, Guangzhou, China. Her research interests include distributed synchronization algorithms in dynamic systems and wireless ad hoc environments. She has published several papers on mutual exclusion problems.

Jiannong Cao received the BSc degree in computer science from Nanjing University, China, in 1982, and the MSc and PhD degrees in computer science from Washington State University, in 1986 and 1990, respectively. He is currently the head and chair professor of the Department of Computing at Hong Kong Polytechnic University, Hong Kong. His research interest includes parallel and distributed computing, networking, mobile and wireless computing, fault tolerance, and distributed software architecture. He has published more than 200 technical papers in the above areas. His recent research has focused on mobile and pervasive computing and mobile cloud computing. He has served as a member of editorial boards of several international journals, a reviewer for international journals/ conference proceedings, and also as an organizing/program committee member for many international conferences. He is a senior member of the China Computer Federation, a senior member of the IEEE, and a member of the ACM.