Reliable Neighborcast Protocol for Vehicular Ad hoc Networks

Patcharinee Tientrakool

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2011

ABSTRACT

Reliable Neighborcast Protocol for Vehicular Ad hoc Networks

Patcharinee Tientrakool

This dissertation introduces a new communication paradigm, neighborcast, for vehicular ad hoc networks and proposes a new communication protocol, reliable neighborcast protocol (RNP), to implement the paradigm. Vehicular applications such as collision avoidance can benefit from allowing vehicles to communicate with their nearby vehicles in order to coordinate movements. Neighborcast is a new paradigm for communications between each vehicle and all nearby vehicles that are within a specified distance from it i.e., its neighbors. In neighborcast, each vehicle has its own set of vehicles with which it wants to communicate i.e., the set of its neighbors, which is different from that of other vehicles. Our proposed communication protocol, RNP, is aimed at providing reliable neighborcast communications. It provides guaranteed message delivery from each vehicle in a vehicular ad hoc network to all of its neighbors within a bounded delay, ensures that all the neighbors that receive the same messages sequence them in the same order and use each of them at the same time, and provides the neighbors the knowledge of whether all of the other neighbors have received the message or which neighbors are missing the message.

The implementation of RNP is significantly different from reliable multicast/broadcast protocols. In a reliable multicast/broadcast protocol, all

communicating vehicles are in one group. But in our RNP, the group size is constrained to limit the communication delay, so we cannot have all vehicles in one group. As a result, we organize vehicles into several overlapping groups and each vehicle may communicate in more than one overlapping group.

RNP is created as an overlay protocol on top of overlapping broadcast groups that use a modified version of a recently invented reliable broadcast protocol, M-RBP, and transfers the guarantees provided by the modified M-RBP from the broadcast group level to the neighborhood level. RNP is composed of two parts. The first is the self-organizing protocol that organizes vehicles into overlapping broadcast groups that use the modified version of M-RBP. The self-organizing protocol ensures that each vehicle is always a member of at least one broadcast group containing itself and all of its neighbors. This way, it can reach all of its neighbors by transmitting messages in one broadcast group, resulting in the same message sequencing for all neighbors. The self-organizing protocol also limits the size of each broadcast group to limit the message delivery delay, limits the number of broadcast groups of which a vehicle is a member to limit the number of recovery messages, and moves the broadcast groups with the vehicles to limit the rate at which a vehicle changes groups. The second part of RNP is the mechanism that transfers the guarantees from M-RBP to provide the RNP guarantees.

In this dissertation, we also show an example of using RNP in conjunction with sensors to avoid rear-end collisions. We propose a simple set of rules for using RNP with sensors to automatically maintain a safe following distance, provide warnings of emergency situations, and negotiate the safe deceleration rates among nearby

communicating vehicles. We quantify the highway capacity improvement from using RNP and compare it with that of using sensors alone.

Table of Contents

List of Figures

Acknowledgments

This thesis would not have been possible without the sponsorship from CAT Telecom Public Company Limited. Thank you for giving me the chance to pursue both my master's degree and Ph.D. at Columbia University.

I am heartily thankful to my advisor, Prof. Nick F. Maxemchuk, for his insightful criticisms, advice, kindness, and patient encouragement. I have learned a lot from him, from critical thinking and systematic project management to academic writing. He always had time for me and has guided me through every step of my Ph.D. study. It is my honor to have worked with him.

I would like to thank Sing Wang Ho and YaChi Ho for their help with the coding. I am glad to have been given the chance to work with both of them. I also thank my office mates Maulik Desai, Kyung Wook Hwang, and Rob Turetsky for their support and for always making me feel that I was not alone in my studies.

I am truly grateful for my family. Their love and support got me through difficult times and enable me to finally complete this thesis. I am thankful to my boyfriend, Tosaporn Chaiwong, who is always by my side and teaches me how to be stronger and to handle all the emotional attacks during the study.

I would like to thank my friend Auranuch Lorsakul for her support and help with printing cool, crisp figures in my dissertation. Finally, I thank all my friends in New York and Thailand. I am glad to have them all in my life.

This thesis is dedicated to my parents, Patcharee and Arthorn Tientrakool,

for their love, endless support, and encouragement.

Chapter 1

Introduction

The objectives of this dissertation are to explore a new communication paradigm for vehicular ad hoc networks called neighborcast and to develop a communication protocol called reliable neighborcast protocol (RNP) to implement the paradigm. Vehicular ad hoc networks (VANETs) are a type of mobile ad hoc network (MANET) formed by vehicles to allow them to exchange information with each other. There are various VANET applications, but we are interested in applications that improve road safety by avoiding or mitigating road accidents. These applications can benefit from allowing vehicles to communicate with their nearby vehicles in order to coordinate their actions. In this dissertation, we will introduce neighborcast, which is a new paradigm for communications between each vehicle and all other vehicles within a specified distance from it i.e., its neighbors, describe a new communication protocol, RNP, which provides reliable neighborcast communications, and show an example of applying RNP to avoid rear-end collisions.

RNP is aimed at providing the following guarantees. It provides guaranteed message delivery from each vehicle in a VANET to all of its neighbors within a bounded delay, ensures that all the neighbors that receive the same messages sequence them in the same order and use each of them at the same time, and provides the neighbors the knowledge of whether all of the other neighbors have received the message or which neighbors are missing the message.

Neighborcast can be considered as a special case of multicast where each vehicle communicates with a subset of nearby vehicles; however, the implementation of reliable multicast/broadcast protocols and our reliable neighborcast protocol, RNP, are significantly different. In a reliable multicast/broadcast protocol, all communicating vehicles are in one group. But in our reliable neighborcast protocol, the group size is constrained to limit the communication delay, so we cannot have all vehicles in one group. As a result, each vehicle may communicate in more than one overlapping group.

RNP is based on a modified version of a recently invented communication protocol called mobile reliable broadcast protocol (M-RBP). M-RBP is one of the protocols developed for mobile ad hoc networks. It provides reliable communications between each member and all other members in the same broadcast group and also provides a set of guarantees that is very useful for road safety applications, including guaranteed message delivery and bounded message delivery delay. Because of the useful set of guarantees that M-RBP provides and the similarity between the reliable broadcast communications provided by M-RBP and the reliable neighborcast communications that we want, we have created RNP in a way that utilizes a modified version of M-RBP to achieve reliable neighborcast rather than creating it as an entirely new communication protocol. More details of M-RBP and the modified version of M-RBP used in this dissertation are provided in chapter 2.

Our contribution is to create RNP as an overlay protocol on top of broadcast groups that use the modified version of M-RBP. Because the modified version of M-RBP provides reliable communications between each member and all other members in the same broadcast group, if we can put each vehicle and all of its neighbors in the same

broadcast group, then we can achieve the reliable neighborcast communications that we want. Therefore, RNP needs to have a mechanism that organizes vehicles into broadcast groups as well as a mechanism to transfer the set of guarantees provided by the modified version of M-RBP from the broadcast group level to the neighborhood level in order to provide reliable neighborcast communications.

As the first part of RNP, we have created a self-organizing protocol that organizes vehicles into overlapping broadcast groups that use the modified version of M-RBP. Grouping vehicles into broadcast groups is not straightforward. One challenge is that the message delivery delay provided by M-RBP increases with the number of members in a broadcast group, so we need to limit the size of the group in order to limit the delay. Broadcast groups must overlap and each vehicle must be a member of all the groups that overlap its location so that each vehicle can communicate with all of its neighbors. Since each vehicle has to recover all messages from every group of which it is a member, the number of recovery messages increases with the number of groups to which it belongs. Therefore, we also need to limit the number of groups of which each vehicle is a member. We have developed a self-organizing protocol that meets all of these requirements. The self-organizing protocol ensures that each vehicle is always a member of at least one broadcast group containing itself and all of its neighbors, while simultaneously limiting the size of each broadcast group, limiting the number of broadcast groups to which a vehicle belongs, and moving the broadcast groups with the vehicles to limit the rate at which a vehicle changes groups. The details of the self-organizing protocol are presented in chapter 3.

We evaluate the performance of the self-organizing protocol in terms of the average number of times that a vehicle joins a new group per minute and the average number of groups of which a vehicle is a member. The performance evaluation is determined by analysis and the results are verified with simulations. The average number of times that a vehicle joins a new group per minute from our self-organizing protocol is also compared with stationary groups. The detail of the performance of the self-organizing protocol is presented in chapter 4.

As the second part of RNP, we have created a mechanism to provide reliable neighborcast communications. Since our goal is to provide guaranteed message delivery between each sending vehicle and all of its neighbors, not between each sending vehicle and all other group members, it is inefficient to have all group members recover messages transmitted to the group as in M-RBP. Therefore, the mechanism allows only the neighbors of each sending vehicle to recover and use the message and also transfers the guarantees provided by the underlying protocol, the modified version of M-RBP, from the broadcast group level to the neighborhood level. This mechanism is described in chapter 5.

We show the performance of RNP in terms of 1) the maximum delay until all neighbors of a sending vehicle successfully receive and commit an application-level message from the sending vehicle, and 2) the average number of messages occurring from one application-level message transmission. The latter metric is compared with a simple ARQ protocol that allows each sending vehicle to repeatedly transmit its message until it receives an acknowledgement from each of its neighbors. The performance

evaluation is determined by analysis, and the results are verified with simulations. The detail of the performance of RNP is described in chapter 6.

Finally, we show an example of using RNP in conjunction with sensors to avoid rear-end collisions and quantify the highway capacity improvement in chapter 7. We vary the percentage of vehicles equipped with both sensors and RNP to study its effects on highway capacity. We also compare the capacity improvement for using RNP with sensors with using sensors alone.

In this dissertation, we make the following contributions. We propose a new communication paradigm called neighborcast, a new communication protocol for VANETs called reliable neighborcast protocol (RNP), a self-organizing protocol which organizes vehicles into moving broadcast groups, and the mechanism that transfers the set of guarantees from the broadcast group level to the neighborhood level to provide reliable neighborcast communications. We evaluate the performance of the self-organizing protocol in terms of the average number of groups of which a vehicle is a member and the average number of times that a vehicle joins a new group per minute and compare this latter metric with a stationary group approach. We evaluate the performance of RNP in terms of the maximum delay until all neighbors successfully receive and commit an application-level message and the average number of messages occurring from one application-level message transmission and compare the average number of messages occurring with the simple ARQ protocol. We show an example of using RNP with sensors to avoid rear-end collision, quantify the percentage increase in highway capacity that RNP can provide, and compare the advantages of using RNP with sensors

over using sensors alone. This example provides a basic understanding of highway capacity benefits that communication protocols can provide while improving road safety.

In this chapter, we describe the background of VANETs and VANET applications in section 1.1 and provide related work on communication protocols for VANETs in section 1.2.

1.1 Background

This section provides background information about vehicular ad hoc networks (VANETs). We explain what VANETs are, describe VANET applications, and discuss the requirements on communication protocols for VANETs.

1.1.1 VANETs definition

Vehicular ad hoc networks (VANETs) are a type of mobile ad hoc networks (MANETs) that is formed by vehicles equipped with wireless communication devices. These networks do not rely on fixed infrastructure and the network topologies constantly evolve. VANETs allow vehicles to exchange information with each other. The information may range from vehicle motion data to Internet media content, depending on the application.

1.1.2 VANET applications

There are several applications of VANETs. These applications can be categorized into 4 types based on general aim [1] as follows:

Type 1: General information services

This type of application [2-8] provides information services for vehicles on public roads. The information source may be a vehicle in the network or lie outside of the network, such as in the wired Internet, and the information may be propagated extensively. The provided information does not involve vehicle safety e.g., available parking spots, advertising, traffic information, and entertainment feeds.

Type 2: Vehicle safety information services

This type of application [9-20] provides information services similarly to Type 1 applications but the provided information is safety-related such as accident warnings, road conditions, and obstacle warnings. However, the provided information is not used to automatically control vehicles.

Type 3: Individual motion control

Type 3 applications [21-26] avoid collisions between vehicles by allowing vehicles to automatically control their individual motions by using information sensed from their sensors and gathered from their neighboring vehicles. No group motion coordination is performed. An example of this type of application is adaptive cruise control [21, 22], where a vehicle automatically adjusts its own speed to maintain a safe following distance with the preceding vehicle based on the speed, acceleration, and fault conditions of the preceding vehicle received from communications with the preceding vehicle. Another example is aircraft collision avoidance [23] that detects, prevents, and resolves airborne conflict by using information received from other aircraft within range.

Type 4: Group motion control

Type 4 applications [27-34] involve group motion coordination among vehicles in order to avoid collisions between vehicles and increase the capacity of a highway. An example is vehicle platooning [28-33], where vehicles on a highway are organized in platoons. The vehicles communicate with their neighboring vehicles and use distributed control techniques in order to travel in close proximity to one another without colliding. [27] describes a cooperative driver assistance application for highway merging. [34] describes cooperative driving to avoid collisions at blind crossings. Vehicles enter the crossing area in small groups. When two vehicles from different groups communicate with each other, they exchange the information of all vehicles in their groups. By using the information from the vehicles approaching the crossing area, all possible safety driving plans can be determined. A safety driving plan specifies an order of vehicles crossing the intersection that does not cause collision.

1.1.3 Requirements on communication protocols for VANETs

Each type of application establishes its own requirements on communication protocols in terms of message delivery delay, message delivery reliability, and communication scope.

Type 1 applications are the only applications that are not related to road safety so they can tolerate message delivery delay and still operate correctly. They may tolerate a best effort message delivery service with intermittent communication failures, such as loss of a query response or damaged media frames. This type of application requires messages to be broadcast throughout a large area.

Type 2 applications have a strict message delivery delay requirement since they use communications to ensure safe separation and they operate essentially blind to hazards when warning messages are delayed. These applications rely on highly probable message delivery and need to fall back to alternate control if too many packets are lost. They also require messages to be broadcast throughout a large area as in Type 1 applications.

Type 3 applications have a strict message delivery delay requirement and rely on highly probable message delivery as in Type 2 applications. They typically require localized communication among nearby vehicles.

Type 4 applications can be varied. Some applications, such as platooning, require strict message delivery delay; while in some applications, such as intersection collision avoidance, delayed messages just lead to delayed use of the intersection but do not cause a failure. This type of application requires the additional ability to determine whether messages are received by the intended receiving vehicles in order for the movement coordination among these vehicles to be successful. With this level of service, the vehicles can take alternate action if message delivery is not confirmed by a deadline. This type of application requires localized communication among nearby vehicles.

1.2 Related work on communication protocols for VANETs

Although many of the requirements of VANET applications generally apply to MANETs, many MANET applications will tolerate lost or delayed information, whereas these may be catastrophic events for VANET applications involving vehicle safety or

motion control. In addition, due to high node mobility in VANETs, communication protocols developed for MANETs might not perform well in VANETs. Several communication protocols have been developed specifically for VANETs. This section presents related work on communication protocols for VANETs.

Numerous broadcast protocols have been developed for information dissemination in VANETs [2-6, 9-15, 17-19, 35, 36], which can be used for Type 1 and Type 2 applications. The simplest approach is broadcast flooding that uses a flat organization and allows each vehicle to forward a copy of each new message once to all one-hop peers. Since the communication follows a mesh instead of a tree, it offers many redundant message paths and trades increased bandwidth utilization for improved connectivity. Forwarded messages can be assembled with others and retransmitted in a single packet for more efficient use of the communication channel [35]. Retransmitters can be selected using a relay algorithm that reduces the number of broadcasts while maintaining radio coverage. For example, a vehicle may choose not to rebroadcast a periodic alert if it overhears a peer farther from the source doing so [9]. The solution in [12] adjusts the probability of packet forwarding and the time that each vehicle has to wait before forwarding packets based on the number of its one-hop and two-hop neighbors. Vehicles that can cover a greater number of 2-hop neighbors are given higher forwarding probability, which leads to shorter delay time to forward packets. [36] proposes the TRAcking DEtection (TRADE) relay algorithm, in which peers are interrogated for their locations and driving contexts (i.e., road, direction of travel, etc.) before relays are appointed. Messages then identify the intended relay vehicles and the context to use for subsequent forwarding.

Broadcast protocols may use opportunistic flooding, which is a technique used in delay-tolerant and intermittently connected networks [37], for packet dissemination. In opportunistic flooding, a vehicle temporarily stores messages when a network partition is encountered and waits for opportunities to forward them at a later time. Messages that remain relevant are forwarded when a vehicle that will move the messages closer to their destination is encountered. The regional alert system (RAS) in [11] uses opportunistic flooding to pass a token containing an accident alert between vehicles in the presence of temporary partitions. The token can also be passed to cars moving further away from the location of the accident in the opposite lane to help reduce the problem in a low-density road, where alerts cannot propagate because there is no car in the same lane to which the token can be passed. The mobility-centric data dissemination protocol (MDDV) described in [38] combines opportunistic flooding with geocast-based trajectory forwarding. The objective of MDDV is to forward a message along a trajectory to a geographical destination region as fast as possible while dealing with rapidly changing and partitionable VANET topologies.

Vehicles can be organized hierarchically into groups or clusters to reduce the amount of rebroadcast messages. In [10], the road is divided into broadcast cells with equal length that move with the vehicles. One vehicle serves as a cell reflector of each cell and is responsible for relaying messages to neighboring cells and broadcasting messages to all vehicles in its cell.

Another important purpose of organized communication is to reliably and efficiently deliver messages to an intended receiver group. This is of particular importance for Type 3 and Type 4 applications. [39] proposes the local peer groups

(LPGs) to support communications among neighboring vehicles. The LPGs can be stationary or dynamic. In stationary LPGs, a GPS-based grid is used to partition the road into stationary and well-defined LPGs. Members of LPG dynamically change as vehicles move. While in dynamic LPGs, nearby vehicles dynamically form an LPG group. MAC-level communication can be used for intra-LPG communication to tightly coordinate the motion of nearby vehicles. [40] proposes the application level clustering concept, in which each application e.g., platooning, a highway merge assistant, sets up its own virtual clusters. A temporary cluster controller (CC) is selected for each cluster and is responsible for collecting messages for a specific application from vehicles in the cluster, processing them, and disseminating them to all vehicles in the cluster. The use of CC ensures data consistency and reliable communication. [41] describes the Wireless Token Ring Protocol (WTRP), which can be used for platooning application. It is a token-based scheme to support rapid, periodic communication among vehicles within a platoon. A vehicle receives the token, transmits its data, and explicitly passes the token to a one-hop neighbor in its ring. Another protocol for communication within a group of vehicles is the Mobile Reliable Broadcast Protocol (M-RBP) [42,43]. The protocol is developed for MANETs but can also be used for VANETs. It provides a message recovery and voting process that guarantees message delivery to all members in a broadcast group and ensures that all members use the same messages at the same time.

In this dissertation, we utilize a modified version of M-RBP and create our reliable neighborcast protocol (RNP) as an overlay protocol on it. Therefore, more details of M-RBP and the modified version that we use will be described in section 2.3.

Chapter 2

Goals and concepts

This chapter describes the goals of the dissertation and the concept behind the development of our communication protocol, reliable neighborcast protocol (RNP). In section 2.1, we specify the type of VANET applications on which this dissertation is focusing and describe the goals of this dissertation. We introduce a new communication paradigm called neighborcast and the concept of neighborhood, which is the basis of RNP in section 2.2 and describe a reliable broadcast protocol, M-RBP, which is the protocol on which our RNP is based in section 2.3.

M-RBP is not part of our contribution of this dissertation but we use a modified version of it in this dissertation since it provides the guarantees that are useful for RNP. The token passing mechanism, the recovery and voting mechanism, and the joining process of M-RBP are important to us so they are described in section 2.3.1, 2.3.2-2.3.3, and 2.3.4 respectively. These mechanisms of M-RBP are described in detail in order to provide a background for understanding the operation of RNP. The modified version of M-RBP used in this dissertation is described in section 2.3.5 and the important guarantees that it provides are described in section 2.3.6.

Lastly in this chapter, we present the concept of how to create RNP as an overlay protocol on top of the modified version of M-RBP and describe the assumptions that we use throughout the dissertation and for creating RNP in section 2.4.

2.1 Goals of the dissertation

In this section, we describe the goals of the dissertation and state the type of VANET applications on which we are focusing. We explain why existing reliable broadcast protocols cannot be used to achieve our goals, and why we need to create a new communication protocol, RNP.

The goals of this dissertation are to explore a new communication paradigm for VANETs called neighborcast and to develop a communication protocol called reliable neighborcast protocol (RNP) to implement the paradigm. We are interested in VANET applications that improve road safety by avoiding collisions or, specifically, Type 3 and Type 4 applications described in section 1.1.2. Collisions can be avoided more effectively when vehicles are allowed to communicate with nearby vehicles to coordinate their movements. Neighborcast is a new paradigm for communications between each vehicle and all nearby vehicles that are within a specified distance from it i.e., its neighbors. The detailed description of neighborcast and neighborhood concept is provided in section 2.2.

Existing reliable broadcast protocols cannot be directly used to provide reliable neighborcast communications. Existing reliable broadcast protocols provide reliable message delivery to all members in a broadcast group, but in reliable neighborcast, we need reliable message delivery to nearby vehicles around each sending vehicle. The group of nearby vehicles of a sending vehicle is distinct from those of other sending vehicles, so communications is not contained in a well-defined group of members as in the case of broadcast, but is spread over a large area. Therefore, we need to create a new communication protocol to provide reliable neighborcast communications.

We have created a new communication protocol, RNP, to provide reliable neighborcast communications. RNP is aimed at providing guaranteed message delivery from each vehicle in a VANET to all of its neighbors within a bounded delay, ensuring that all the neighbors that receive the same messages sequence them in the same order and use each of them at the same time, and providing the neighbors the knowledge of whether all of the other neighbors have received the message or which neighbors are missing the message.

As an example, we have applied RNP in conjunction with sensors to avoid rear-end collisions. We have quantified the highway capacity improvement that it can provide and compared the capacity improvement with the use of sensors alone. The details of this will be presented in chapter 7.

2.2 Neighborcast and neighborhood concepts

In this section, we provide the definitions of neighbors, neighborhood, and neighborhood span, and describe a new communication paradigm, neighborcast. The difference between neighborcast and broadcast/multicast is also described.

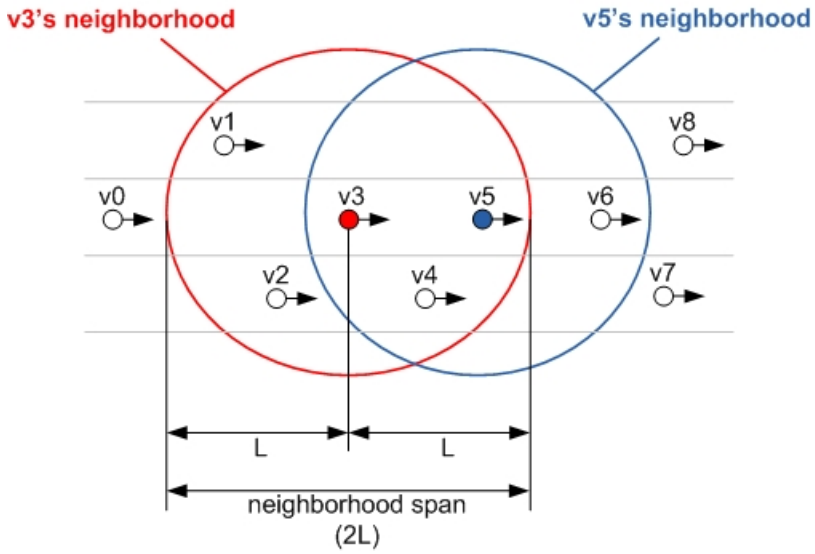

Figure 1. Neighborhood and neighborhood span

The "neighbors" of a vehicle are defined as all vehicles within a specified distance L from that vehicle. The area that covers the distance L around the vehicle is called the vehicle's "neighborhood" and length 2L is the "neighborhood span". In Figure 1, the red circle is centered at the position of vehicle 3 and has a radius of L. The area covered by the red circle is vehicle 3's neighborhood. Since all vehicles in the red circle are within distance L from vehicle 3, they are vehicle 3's neighbors. Specifically, vehicle 1, 2, 4, and 5 are vehicle 3's neighbors.

"Neighborcast" is a new communication paradigm that is useful for VANETs. It is defined as the communications between each vehicle in the network and all of its neighbors. In neighborcast, each vehicle has its own set of vehicles with which it wants to communicate i.e., the set of its neighbors, which is different from that of other vehicles. Neighborcast is especially useful for applications that require information exchange between each vehicle and its neighboring vehicles in order to coordinate movement.

Neighborcast can be considered as a special case of multicast where each vehicle communicates with a subset of nearby vehicles; however, the implementation of reliable multicast/broadcast protocols and our reliable neighborcast protocol, RNP, are significantly different. In a reliable multicast/broadcast protocol, all communicating vehicles are in one group. But in our reliable neighborcast protocol, the group size is constrained to limit the communication delay, so we cannot have all vehicles in one group. As a result, each vehicle may communicate in more than one overlapping group.

2.3 Original M-RBP and the modified version of M-RBP used to create RNP

This section describes a recently invented reliable broadcast protocol for mobile ad hoc networks called mobile reliable broadcast protocol (M-RBP) [42,43]; which is the protocol on which our RNP is based. M-RBP is not part of our contribution of this dissertation. However, we use a modified version of M-RBP in this dissertation because it provides useful guarantees. M-RBP provides many guarantees, but in this section, we will only describe the guarantees that are important to RNP i.e., guaranteed message delivery to all of the receivers in a broadcast group within a bounded delay, and ensuring that all the receivers in the group commit the same message at the same time.

First, in section 2.3.1 to 2.3.4, we will describe the mechanisms and process of M-RBP that are important to us. Next, in section 2.3.5, we will describe the modified version of M-RBP that is used in this dissertation. In section 2.3.6, we will summarize the guarantees from the modified M-RBP that are useful for RNP.

The token passing mechanism is the first mechanism of M-RBP that is important to us. The token passing mechanism permits only one receiver in the broadcast group to acknowledge received source messages at a time. This mechanism also allows the protocol to continue to operate when the members of the group change, which is a necessary characteristic of communication protocols for high node mobility environment in VANETs. Section 2.3.1 describes the token passing mechanism in detail.

The recovery and voting mechanism and the joining process of M-RBP are also important to us. M-RBP uses the token/source message recovery and voting mechanism to ensure that all the receivers in the group commit the same token/source message at the same time, as well as to keep track of the current receivers in the group. The token recovery and voting mechanism and the source message recovery and voting mechanism are described in detail in section 2.3.2 and 2.3.3, respectively. The joining process is the process of a receiver joining a new broadcast group and is described in section 2.3.4.

2.3.1 Aggressive token passing mechanism

M-RBP is a reliable broadcast protocol that uses periodic token passing among the receivers in a broadcast group to allow only one receiver to be the token site and acknowledge received source messages at a time. In M-RBP, m mobile units in the broadcast group can serve as both message sources and receivers as shown in Figure 2. A token is passed among the m receivers in the group every Δ_T seconds. Every Δ_T seconds, only one receiver is scheduled to transmit the token and is referred to as the token site. The token site is responsible for acknowledging all source messages that it receives during the period Δ_T seconds before its scheduled token transmission time. It

acknowledges received source messages by including an ACK that references all the acknowledged source messages and assigns globally unique sequence numbers for the acknowledged messages in its scheduled token.

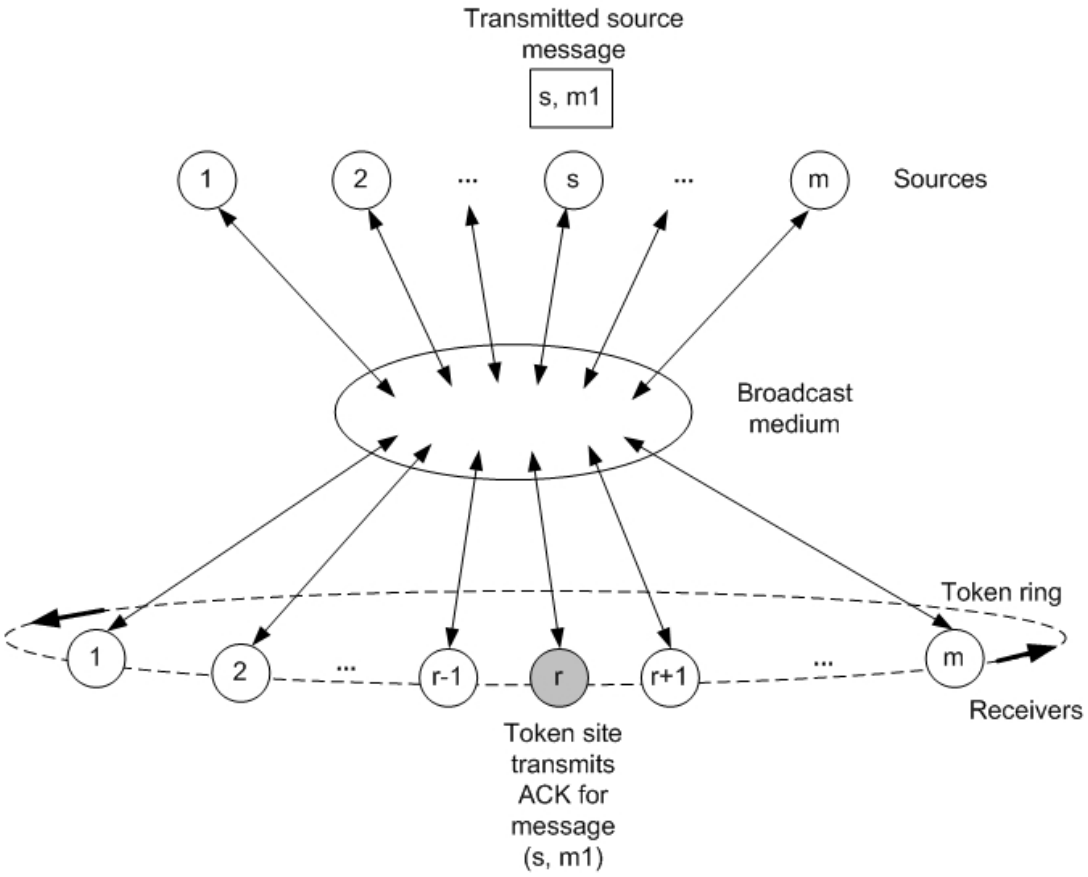

Figure 2. Mobile Reliable Broadcast Protocol (M-RBP) [42]

M-RBP's periodic token passing uses an aggressive token passing mechanism to allow the protocol to continue to operate even when the group changes. This is an important characteristic of communication protocols for networks with node mobility, such as mobile ad hoc networks and VANETs. Since the token is scheduled to be passed to the next token site every Δ_T seconds, the next scheduled token site can still transmit the token at its scheduled time even if the previous token site has left the group and did not transmit the token at its scheduled time.

2.3.2 Token recovery and voting mechanism

In M-RBP, all the receivers in the group recover and vote for the tokens in order to decide whether or not to commit each token. All receivers expect to receive the token from a token site at its scheduled transmission time. If a receiver does not receive any token at the scheduled transmission time, it will recover the missing token by transmitting a negative acknowledgement (NACK) to request a token retransmission. After trying to recover the token, each receiver votes whether or not it has received the token by including its vote in its scheduled token. When the vote is complete, if the majority of receivers voted that they received the token, then the token will be committed, and the ACK included in that token will be used to sequence the acknowledged source messages. On the other hand, if the majority of the receivers voted that they had not received the token, then the token will not be committed and the ACK included in that token will not be used for source message sequencing.

Token voting helps keep track of the current receivers in the group. It is used to determine whether or not the token site that was scheduled to transmit the token has left the group. If the majority of the receivers voted that they had not received the token from the scheduled token site, then all of them will assume that the token site did not transmit the token at its scheduled time because it had already left the group by that time. All the receivers will remove that token site from the group.

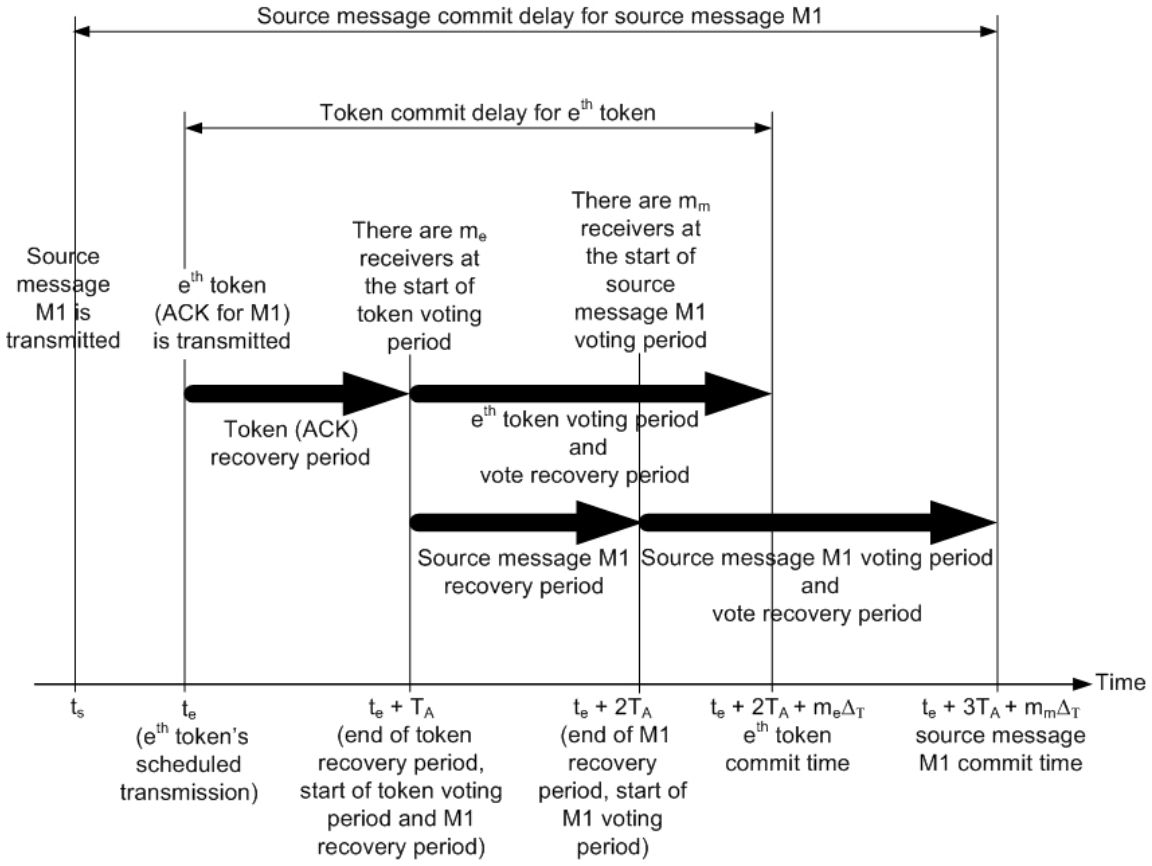

Figure 3. M-RBP timeline

The timeline for the M-RBP recovery and voting process is shown in Figure 3. A source message M1 is transmitted at time t_s. The e^{th} token, which includes the ACK for message M1, is scheduled to be transmitted at time t_e. The receivers in the group expect to receive the e^{th} token at time t_e and begin a recovery process for the token if they do not receive it shortly after t_e. The maximum allowed recovery time for the token, is $T_A = n_{max}T_R$, where T_R is the time between recovery iterations, and n_{max} is the maximum number of recovery iterations allowed. The e^{th} token recovery period ends at time $t_e + T_A$. After the end of the token recovery period, the token voting period starts and the receivers vote whether they have received the e^{th} token or not by including their votes in their

scheduled tokens when their turns to transmit the token arrive. If there are m_e receivers in the group at the start of the token voting period, then by the time $t_e + 2T_A + m_e\Delta_T$, all of the receivers in the group will have voted for the e^{th} token and the tokens containing the votes will have been recovered by the receivers in the group. The vote for the e^{th} token is then tallied at each of the receivers at $t_e + 2T_A + m_e\Delta_T$ (called the token commit time) and each receiver makes the decision whether to commit the e^{th} token or not and whether the token site that was scheduled to transmit the e^{th} token has left the group at this time.

The vote tallying is distributed. Each receiver has to tally the vote based only on the votes that it has received. Each receiver may not receive all of the votes from all the receivers in the group. In other words, receiver r_j receives $A_j\ (e) + B_j\ (e)$ votes from the total $A(e) + B(e)$ transmitted votes, where

$$A_j\ (e) \leq A(e) \quad \text{and} \quad B_j\ (e) \leq B(e)$$

and

$A(e)$ is the total number of "YES" votes transmitted from all the receivers that have received the e^{th} token;

$B(e)$ is the total number of "NO" votes transmitted from all the receivers that have not received the e^{th} token;

$A_j\ (e)$ is the total number of "YES" votes that receiver r_j has received;

$B_j\ (e)$ is the total number of "NO" votes that receiver r_j has received.

Each receiver r_j uses the following rules to make the decision whether to commit the e^{th} token or not and whether to remove the token site that was scheduled to transmit the e^{th} token from the group.

- If $A_j(e) > m_e/2$, then $A(e) > m_e/2$, so r_j leaves the token site that was scheduled to transmit the e^{th} token in the group and commit the e^{th} token.
- If $B_j(e) \geq m_e/2$, then $B(e) \geq m_e/2$ and $A(e) \leq m_e/2$, so r_j removes the token site that was scheduled to transmit the e^{th} token from the group and does not commit the e^{th} token.
- If $B_j(e) < m_e/2$, and $A_j(e) \leq m_e/2$, then r_j is uncertain whether or not $A(e) \leq m_e/2$, so r_j leaves the group itself.
- If $A_j(e) > m_e/2$, but r_j has not recovered the e^{th} token, then r_j leaves the group itself since the other receivers decide to commit the e^{th} token but r_j does not have the e^{th} token.

(Note that m_e is the number of receivers in the group at the start of the e^{th} token's voting period i.e., at time $t_e + T_A$)

There is a token commit delay until a token is successfully committed. The token commit delay is the interval from the time that the token is transmitted by the token site to the time that the token is committed (token commit time). The token commit delay depends on the token passing interval Δ_T, the maximum allowed recovery time for the token T_A, and the number of receivers in the group. The token commit delay of the e^{th} token is shown in Figure 3.

2.3.3 Source message recovery and voting mechanism

All receivers in the group also perform similar recovery and voting process for the source messages acknowledged by the e^{th} token in order to decide which source messages will be committed. The recovery period for the source messages acknowledged by the e^{th} token begins after the end of the e^{th} token recovery period i.e., at time $t_e + T_A$ in Figure 3. At this time, if a receiver has the e^{th} token, then it knows the list of the source messages acknowledged by the token and begins recovering missing source messages. The source message recovery period ends at time $t_e + 2T_A$ and the voting period for each acknowledged source message starts at this time. Each receiver tallies votes for each source message in a distributed manner at the source message commit time $t_e + 3T_A + m_m\Delta_T$, where m_m is the number of receivers in the group at the start of the source message voting period. Each receiver uses the same decision rules as in the token voting process to make a decision whether to commit each acknowledged source message or not. However, a small difference is that if a source message failed to be received by the majority of the receivers, the source message will not be committed as in the token case, but the source that transmitted the message will not be removed from the group.

There is a source message commit delay until a source message is successfully committed. The source message commit delay is the interval from the time that the source message is transmitted by the source, to the time that the source message is committed (source message commit time). The source message commit delay depends on the token passing interval Δ_T, the time until an ACK for the source message is transmitted, the

maximum allowed recovery time T_A, and the number of receivers in the group. The source message commit delay of message M1 is shown in Figure 3.

2.3.4 Joining a new broadcast group

A receiver requests to join a new broadcast group by sending a join request message to the group; it is successfully accepted to the group if its join request is voted in. A receiver that wants to join a new group sends a join request as a source message to the group and waits until the vote for its join request message is complete. If the join request message is voted in by majority of the group, then the receiver is accepted to the group and its join attempt is successful. On the other hand, if the join request is not voted in by the majority of the group, then the receiver fails to join the group and needs to send another join request message to try to join the group again.

There is a join delay until a receiver is successfully accepted to a new group. This is the delay until the join request from the receiver is recovered, voted on, and committed by the current receivers in that group. Since a join request is sent out as a source message, the join delay is equal to the message commit delay of the join request. Therefore, the join delay depends on the token passing interval Δ_T of the new group, the time until an ACK for the join request is transmitted, the maximum allowed recovery time T_A, and the number of receivers in the new group.

2.3.5 The modified version of M-RBP used in this dissertation

In this dissertation, we use a modified version of M-RBP instead of the original version. The modified version is exactly the same as the original version except that it uses a different mechanism to sequence received source messages. The modified version does not use ACK from the token to sequence received source messages; instead, the received messages are sequenced based on their message commit times. The modified version has the same token passing mechanism, token recovery and voting mechanism, source message recovery and voting mechanism, and joining process as the original M-RBP. Thus, the timeline for the modified version is the same as the timeline of the original version shown in Figure 3.

2.3.6 Important guarantees provided by the modified version of M-RBP

Despite the difference from the original M-RBP, both the modified version of M-RBP and the original version provide two guarantees that are important to RNP as follows:

1. They guarantee message delivery to all of the receivers in a broadcast group within a bounded delay. Based on the decision rules for committing source messages, if a source message is voted in by the majority of the group, then all the receivers that still remain in the group at the source message commit time must have received and committed the source message. Therefore, both versions of M-RBP guarantee message delivery to all the receivers that are

still in the group by the source message commit time (i.e., within the source message commit delay).

2. They guarantee that all the receivers that receive the same source message commit the message at the same time. Based on the source message recovery and voting mechanism, each receiver will not commit a source message until the source message commit time. Therefore, all the receivers that receive the same source message are guaranteed to commit the message at the same time.

2.4 Creating RNP based on the modified version of M-RBP

In this section, we describe the concept of how to create RNP as an overlay protocol on top of the modified version of M-RBP in order to provide reliable neighborcast communications. We also give an overview of RNP and the assumptions that we use for this dissertation and for developing RNP.

RNP is based on the modified version of M-RBP because it provides guarantees similar to those that RNP wants to achieve. The modified M-RBP provides guaranteed message delivery and ensures that all receivers use the same received message at the same time as RNP needs; however, these guarantees are provided between each member and all other members in the same broadcast group, not between each vehicle and all of its neighbors as we want in RNP. From the similarity between the guarantees that M-RBP provides and the guarantees that we want, we have created RNP in a way that utilizes the

modified version of M-RBP to achieve reliable neighborcast communications rather than creating an entire new communication protocol.

RNP is created as an overlay protocol on top of broadcast groups that use the modified version of M-RBP. Since the modified version of M-RBP provides the needed guarantees between each member and all other members in the same broadcast group, if we can put each vehicle and all of its intended receivers i.e., all of its neighbors, in the same broadcast group, then we can use the modified version of M-RBP within the broadcast group to achieve the needed guarantees at the neighborhood level. Therefore, RNP is designed to be an overlay protocol that runs on top of broadcast groups that use the modified version of M-RBP.

RNP is composed of two parts. The first part is the self-organizing protocol that organizes vehicles into broadcast groups. Its goal is to ensure that each vehicle is always a member of at least one broadcast group that contains itself and all of its neighbors so that each vehicle can transmit messages to all neighbors in one group and the messages will be sequenced in the same order at all neighbors. The self-organizing protocol is described in detail in chapter 3. The second part is the mechanism that provides RNP guarantees. This mechanism transfers the guarantees provided by the underlying protocol (the modified version of M-RBP) from the broadcast group level to the neighborhood level. This part helps us achieve reliable neighborcast communications. It is needed because based on the neighborcast concept, we only need to provide the guarantees to the neighbors of a sending vehicle, not to all vehicles in the broadcast group as in M-RBP. The mechanism is presented in detail in chapter 5.

We use the following assumptions for this dissertation and for creating RNP.

1. Each vehicle has a wireless communication device.
2. Each vehicle knows its position in relation to its preceding and following vehicles.
3. Each vehicle knows its current speed.
4. We assume a one-way highway and a one-dimensional network so all vehicles in the network move in the same direction.
5. Each vehicle can move with a speed between 0 and a specified maximum vehicle speed only. It cannot exceed the maximum speed.
6. Each vehicle has the same neighborhood span of 2L, where L is a specified value.

Chapter 3

Self-organizing protocol

This chapter describes the self-organizing protocol, which is the first part of our Reliable Neighborcast Protocol (RNP). The self-organizing protocol organizes vehicles into overlapping broadcast groups that move with vehicles while ensuring that each vehicle is always a member of at least one broadcast group that contains itself and all of its neighbors. Therefore, each vehicle can transmit messages to all of its neighbors in one group; which results in the same message sequencing at all neighbors.

In this chapter, we first describe the objectives of the self-organizing protocol in section 3.1 followed by our approach to achieve the objectives in section 3.2 and the characteristics of our approach in section 3.3. The overview of the protocol is presented in section 3.4. Finally, the operation details of the protocol, which include the mechanisms to create new groups, join existing groups, move groups with vehicles, leave groups, split a group into two smaller groups, and merge two adjacent groups into a single group are described in section 3.5.

3.1 Objectives

The objectives of our self-organizing protocol are as follows:

1. To ensure that each vehicle is always a member of at least one broadcast group that contains itself and all of its neighbors.

This objective allows each vehicle to transmit messages to all its neighbors in one group, which guarantees that the messages are put in the same order at all neighbors as mentioned previously.

2. To keep the span of each broadcast group small.

   This is to limit the message commit delay. A group with a large span tends to contain a large number of members, which can result in longer time to commit messages as described in section 2.3.3.

3. To keep the number of broadcast groups of which a vehicle is a member small.

   This objective is to keep the number of tokens transmitted by a vehicle and the number of recovery messages for missing tokens and missing source messages small. A vehicle transmits tokens and recovers missing tokens and missing source messages in all groups of which it is a member. Therefore, the number of tokens transmitted by a vehicle and the number of recovery messages increase with the number of groups of which a vehicle is a member.

4. To move the broadcast groups with vehicles.

   This is to reduce the frequency that vehicles change groups, which in turn reduces the number of join request messages and recovery messages for missing join requests. The frequency that a vehicle changes groups increases with the speed of the vehicle in relation to the speed of the group(s) of which it is a member.

3.2 Our approach

We can accomplish the above objectives by creating overlapping broadcast groups where:

1. Adjacent groups overlap by at least a target overlap size (O_T), where the minimum of O_T is the neighborhood span 2L specified in section 2.2. Note that O_T is set to be slightly bigger than 2L to handle the join delay and the delay until the target overlap size is maintained, which will be described in section 3.5.3.
2. Each group does not overlap the center of the adjacent groups.
3. Each group moves with its members.

Figure 4 shows the layout of adjacent groups. C_i is the position of the center of group i, where i = 1,2,...5.

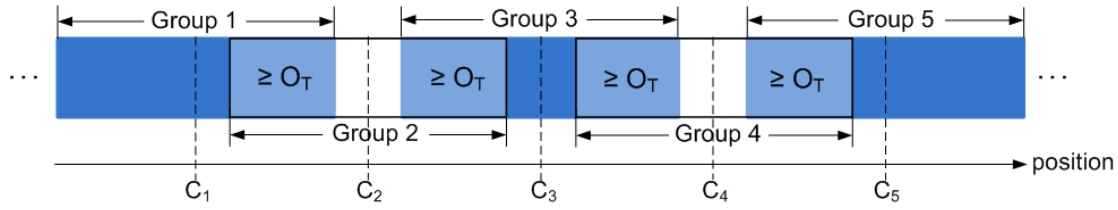

Figure 4. Layout of adjacent groups

3.3 Characteristics of our approach

Our approach has the following characteristics:

1) Each vehicle can reach all of its neighbors in one group

This characteristic derives from the fact that adjacent groups overlap by at least 2L and each group does not overlap the center of the adjacent groups. We show that the

overlap size of at least 2L is both necessary and sufficient for our approach to have all neighbors of a vehicle in one group.

1.1) The overlap size of at least 2L is sufficient for our approach to have all neighbors of a vehicle in one group

All neighbors of a vehicle are in one group as long as the vehicle is at distance ≥ L from both edges of the group. This condition is derived from the definition of neighbors in section 2.2, which states that neighbors of a vehicle are all vehicles within a specified distance L from that vehicle.

Therefore, we show that when our approach is used and the overlap size ≥ 2L, every vehicle is at distance ≥ L from both edges of a group that overlaps its position. We do this by showing that every member of group i in Figure 5 is at distance ≥ L from both edges of a group that overlaps its position, no matter which part of group i it is in.

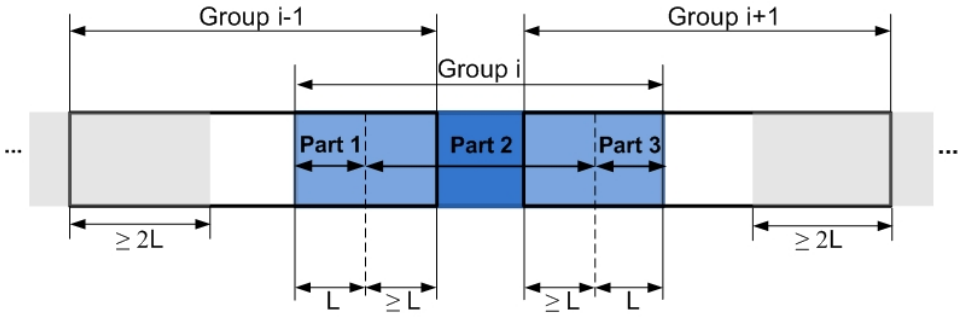

Figure 5. Three overlapping groups constructed according to our approach

The three groups in Figure 5 are constructed according to our approach, and each overlap is ≥ 2L. We can easily see that the members in Part 1 of group i are at distance ≥ L from both edges of group i-1, which overlaps their positions. The members in Part 2 of group i are at distance ≥ L from both edges of group i, which overlaps their positions.

And lastly, the members in Part 3 of group i are at distance ≥ L from both edges of group i+1, which overlaps their positions.

1.2) The overlap size of at least 2L is necessary for our approach to have all neighbors of a vehicle in one group

We show that if the overlap size is smaller than 2L, it is possible that not all neighbors of a vehicle are in one group. In Figure 6, the overlap between groups i and i-1 is smaller than 2L. Vehicle 1 is at distance < L from the trailing edge of group i and < L from the leading edge of group i-1. Thus, neither group i nor group i-1 contains all of vehicle 1's neighbors.

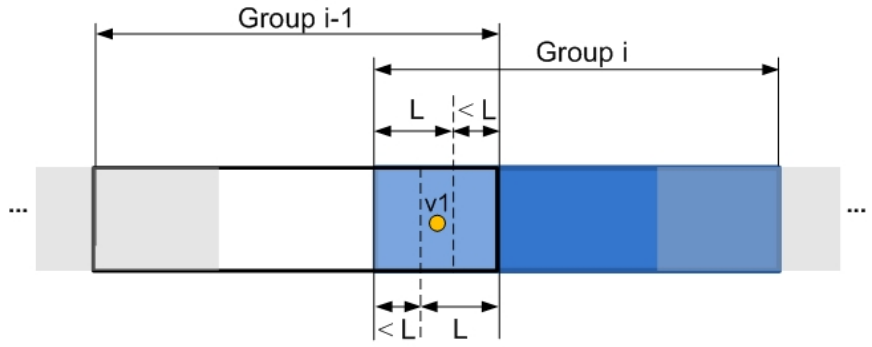

Figure 6. Group i-1 and i overlap by an overlap size smaller than 2L

2) Each vehicle is overlapped by at most two groups.

This characteristic arises from the fact that each group does not overlap the center of the adjacent groups. Every position is overlapped by at most two groups. Hence, each vehicle will be a member of at most two groups. Thus, each vehicle will transmit tokens and recover missing tokens and missing source messages in at most two groups.

3.4 Overview of the self-organizing protocol

This section gives an overview of how the self-organizing protocol operates. We describe the distributed nature of the protocol, the functions performed by each member of a broadcast group, and how we use the token passing and the recovery and voting mechanism of the underlying protocol, the modified version of M-RBP, to ensure that every member performs the same action to the group at the same time.

Each member of a broadcast group is responsible for moving the edges of the group as the members move, splitting the group when the group span becomes too large, merging the group with the adjacent group when the group overlaps the center of the adjacent group, and starting new groups when necessary. The rules for changing the group are independently executed at each member based on proposed changes from group members and the group information locally stored at the member. Each member reports its proposed change to the group when it transmits its scheduled M-RBP token, and the change does not take place until the token is voted in according to the M-RBP voting mechanism as described in section 2.3.2. The voting mechanism ensures that all members always perform the same change to the group at the same time.

The member may also send a source message to the group if it must issue a change when it does not have the token. A source message, which is transmitted by a vehicle in the overlap between two adjacent groups, is used to merge the two groups and simultaneously notify the members of both groups. The source message must also go through the M-RBP voting procedure before the change takes effect.

Each time a vehicle transmits the token, it includes the following information in the token:

- its current position and speed
- proposed edge positions for the group
- current edge positions of the group

Each member records two most recently reported positions and speeds of all group members from the committed tokens so that it can use the information to determine the proposed edge positions of the group during its turn to transmit the token. The proposed edge positions from the tokens are used to move the edges of the group at the token commit times. The current edge positions of the group from the tokens tell other vehicles that are not current members of the group where the group span is, so they can decide whether or not to join the group.

3.5 Operation details

In this section, we describe the operation of the self-organizing protocol in detail. In order to communicate with other vehicles, a vehicle needs to be a member of at least one broadcast group. The vehicle achieves this by either joining existing groups in the network or creating its own group. The details for joining existing groups and creating new groups are described in section 3.5.1.

Each vehicle proposes new edge positions and moves the edges of each group of which it is a member. The processes for proposing new edge positions and moving the edges of the group are described in section 3.5.2. The vehicle splits the group/merges the

group with the adjacent group when the split/merge condition is satisfied. The details for splitting and merging are described in sections 3.5.5 and 3.5.6, respectively. The vehicle leaves the group when the group does not cover its position and some conditions are satisfied. The details for leaving groups are described in section 3.5.4.

Because of the delay for joining a new group and the mechanism to let group members know that they have to maintain the target overlap size with some overlapping group behind them, the target overlap size needs to be extended to be slightly greater than the minimum value of 2L. Section 3.5.3 explains this issue in detail.

3.5.1 Joining existing groups and creating a new group

This section describes how vehicles join existing groups and create new groups. Each vehicle joins all the groups that cover its current position. If the vehicle is not a member of any group and there are no groups that it can join, it creates its own group. When there is a string of groups that ends somewhere and a vehicle is within O_T from the end of the string of groups, the vehicle creates a new group that overlaps beyond the end of the string of groups in case there are neighbors with which it needs to communicate beyond the end of the string of groups.

Each vehicle hears all of the tokens transmitted in its region of the network and joins all the groups whose spans cover its current position. The vehicle follows the joining process of M-RBP described in section 2.3.4 when it wants to join a new group. It can simultaneously join more than one group by simultaneously sending out a join request message to each of the groups. The vehicle is accepted to a new group when its join request message is voted in.

If there are no existing groups that cover the vehicle's position and the vehicle is not a member of any group, it creates its own group with itself at the center of the group and the group span is equal to a target group span (S_T), where $S_T \geq 2O_T$. Then it starts transmitting the token for the group to invite other vehicles to join the group. Figure 7 shows a new group created by vehicle 0. The center of the group is at position x, which is the current position of vehicle 0.

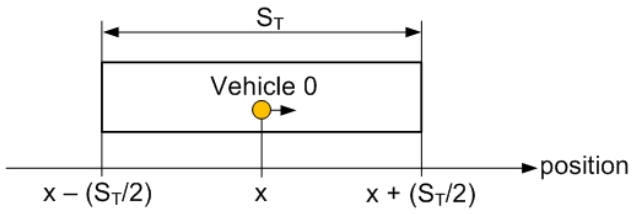

Figure 7. A new group created by vehicle 0

A member of group i that is within the distance O_T from the leading/trailing edge of the group creates a new group in front of/behind group i and overlapping with group i by O_T if there are no other groups in front of/behind group i that cover its current position. The span of the new group is equal to the target group span (S_T). This rule allows a vehicle to create a new overlapping group before it needs to use the new group to communicate with its neighbors. This is in order to give the vehicle's neighbors enough time to join the new group successfully before the vehicle uses the group to communicate with them.

Figure 8 shows a new overlapping group G2 created by vehicle 2, which is within the distance O_T from the trailing edge of group G1, and a new overlapping group G3 created by vehicle 3, which is within the distance O_T from the leading edge of G1.

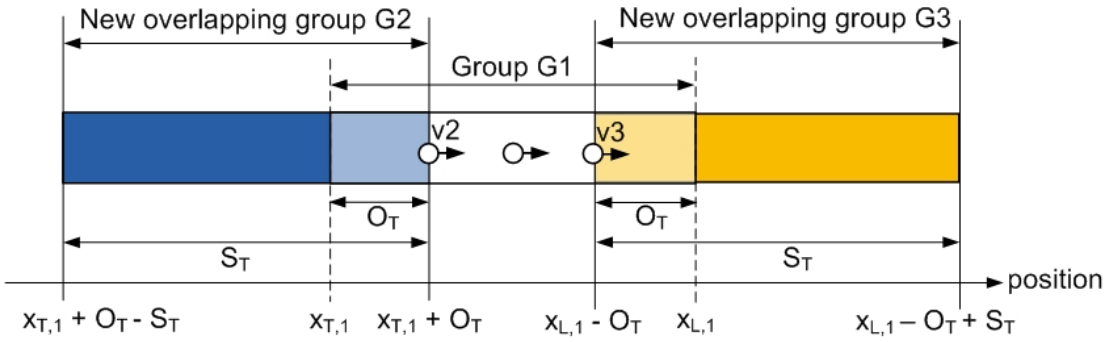

Figure 8. New overlapping groups created by vehicle 2 and vehicle 3

3.5.2 Moving the groups with vehicles

This section describes how to move broadcast groups with vehicles. The goal is to move each group at the median speed of its members, while maintaining the target overlap size O_T with the overlapping group (if any) behind it.

When a single group travels alone, both the leading and trailing edges of the group move at the median speed of the group members. All members are > O_T from the edges. Otherwise, they create a new group overlapping with the current group as described in section 3.5.1.

When two or more groups overlap and are moving together, the leading edge of the group moves at the median speed of the group members; the speed of the trailing edge depends on whether or not there are members in the overlap with the following group. If no members are in the overlap with the following group, the trailing edge moves at the median speed of the group members as if the group is a single group traveling alone. If there is at least one member in the overlap with the following group, the group must maintain the target overlap size O_T with the following group. So, its trailing edge position cannot be > (the leading edge position of the following group - O_T).

Therefore, there are 2 cases of edge movement that we are considering.

Case1: where both edges of the group move at the median speed of the group members.

Case 2: where the leading edge moves at the median speed of the group members, but the trailing edge moves to maintain the target overlap size O_T.

We now provide more details on how each group member calculates the proposed edge positions to be included in its scheduled token in section 3.5.2.1, and how each member determines the new edge positions for the group when a token is committed in section 3.5.2.2.

3.5.2.1 How to calculate the proposed edge positions

Each time a vehicle transmits the token for group i, it proposes the new edge positions for the group. The vehicle uses the same calculation to calculate the proposed edge positions for both cases of edge movement.

The vehicle first estimates the current speed (v_j) of each member j of the group and determines the median speed of the group members (v_G). The calculation of v_j and v_G are as follows:

$$v_j = v_{j,1} + [(v_{j,1} - v_{j,2}) * (t - t_{j,1}) / (t_{j,1} - t_{j,2})] \quad (1)$$

$$v_G = \text{median of all } v_j \quad (2)$$

where

$v_{j,1}$ is the most recently recorded speed of vehicle j.

$t_{j,1}$ is the time that vehicle j had speed $v_{j,1}$.

$v_{j,2}$ is the second most recently recorded speed of vehicle j.

$t_{j,2}$ is the time that vehicle j had speed $v_{j,2}$.

t is the current time (the time that the vehicle transmits the token).

Then the proposed position for the leading edge of group i ($x_{L,p}$) is calculated from v_G as follows:

$$x_{L,p} = x_L + [v_G*(t_c - t)] \quad (3)$$

where

x_L is the current position of the leading edge of group i.

t_c is the estimated commit time of the token.

The proposed position for the trailing edge of group i ($x_{T,p}$) depends on whether the vehicle that transmits the token is a lower overlap vehicle or not. A vehicle is considered a lower overlap vehicle if it is in the overlap between group i and group i-1 behind group i and is a member of both groups.

- If the vehicle that transmits the token is not a lower overlap vehicle,

$$x_{T,p} = \min[\, x_T + [v_G*(t_c - t)] \,,\, x_{L,p} - S_T \,] \quad (4)$$

- If the vehicle that transmits the token is a lower overlap vehicle,

$$x_{T,p} = \min[\, L_{i-1} - O_T \,,\, x_T + [v_G*(t_c - t)] \,,\, x_{L,p} - S_T \,] \quad (5)$$

where

x_T is the current position of the trailing edge of group i.

L_{i-1} is the current position of the leading edge of group i-1 behind group i.

The proposed trailing edge position ensures that the group span will not be smaller than the target group span (S_T) if both proposed leading and trailing edge positions are used to move the group. If the proposing vehicle is a lower overlap vehicle, the proposed trailing edge position also ensures that the overlap between group i and group i-1 will not be smaller than the target overlap size (O_T).

The vehicle also includes in its token whether or not it is a lower overlap vehicle. This information tells other members whether there is any group following group i with which group i needs to maintain the target overlap size or not.

3.5.2.2 How to determine the new edge positions

Each member moves the edges of the group each time a token for the group is committed. The calculation of the new leading edge position is the same for case 1 and case 2 of edge movement. The leading edge is always moved to the maximum position between the current leading edge position of the group and the proposed leading edge position from the token.

The calculations of the new trailing edge position are different for the two cases of edge movement. For case 1, the trailing edge is always moved to the proposed trailing edge position from the token. For case 2, the trailing edge is moved to the proposed trailing edge position from the token only if the token is from a lower overlap vehicle. Otherwise, the trailing edge is not moved.

Each time a token from a lower overlap vehicle is committed, the edges of the group must be moved according to case 2 for $n*\Delta_T$ seconds from the token commit time, where n is the current number of group members and Δ_T is the token passing interval

(i.e., the token is passed every Δ_T seconds). The token reports that there is at least one member, the token sender, in the overlap between the group and some following group. Therefore, the edges of the group must be moved according to case 2 to maintain the target overlap size O_T with the following group. If no other tokens from lower overlap vehicles are committed within the $n*\Delta_T$ second interval, then we assume that no members are in the overlap between the group and the following group anymore. So, the group does not need to maintain the target overlap size anymore and can change to move its edges according to case 1.

3.5.3. Extending the target overlap size to handle the join delay and the delay until group members are informed about the overlap

In this section, we explain the need for extending the target overlap size (O_T) to be greater than the minimum value of 2L to handle the delay for vehicles to successfully join a new group, and the delay until all group members are informed by the vehicle in the overlap that there is some overlapping group behind them and start maintaining the target overlap size with that group. We show that in order to handle these delays, the target overlap size between two adjacent groups must be extended to $2L + 2\Delta + e$ and vehicles must use only the group that they are $\geq L + \Delta$ from both edges of the group to transmit messages to their neighbors.

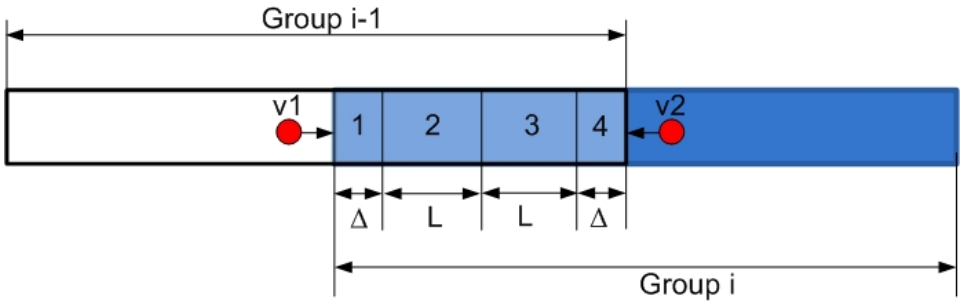

Figure 9. Vehicles v1 and v2 join adjacent groups

A vehicle joins an adjacent group to receive messages that its neighbors transmit in the adjacent group and to transmit messages to its neighbors using the adjacent group when it cannot reach all the neighbors by using the current group. There is a join delay until a join request from a vehicle is recovered, voted, and committed as described in section 2.3.4, so the target overlap size O_T has to be extended to allow vehicles to start joining the adjacent group early enough that their join requests will be committed by the new group before they need to use the group to communicate with their neighbors.

To handle this join delay, the target overlap size must be extended to $2L+2\Delta$, where $2L$ is the neighborhood span and 2Δ is the extra overlap; and vehicles must use only the group where they are $\geq L + \Delta$ from both edges of the group to transmit messages to their neighbors. We now show how this mechanism works. In Figure 9, let the overlap size between two broadcast groups i-1 and i equal to $2L+2\Delta$. The overlap consists of regions 1, 2, 3, and 4. Regions 1 and 4 are Δ long, and region 2 and 3 are L long. Vehicle v1 enters and starts joining group i when it enters region 1. When v1 is in region 1, it can communicate with all its neighbors in group i-1. All v1's neighbors to the front of v1 are in regions 1 and 2, which are $\geq L + \Delta$ from both edges of group i-1 and $< L + \Delta$ from the trailing edge of group i, so they transmit messages in group i-1. When v1 is in region 2,

some of its neighbors are in region 3, which are ≥ L + Δ from both edges of group i and < L + Δ from the leading edge of group i-1, so they transmit messages in group i. To receive messages from these neighbors, v1 must be accepted to group i before it enters region 2. Therefore, v1 has the time it takes to cover distance Δ to join group i. Similarly, when vehicle v2 enters region 4, it has the time it takes to enter region 3 (to cover distance Δ) to join group i-1 successfully before it has to use the group to communicate with its neighbors. The extra overlap 2Δ gives vehicles time to successfully join a new group before they need to use the group for communication. The size of Δ is equal to the join delay times the vehicle speed in relation to the speed of the edge of the group.

The target overlap size, 2L+2Δ, also has to be extended by an additional amount e to handle the time until the member in the overlap informs other group members about the overlapping group behind them and the group members start maintaining the target overlap size with the overlapping group. According to the moving group method described in section 3.5.2, when two groups overlap and there is at least one vehicle in the overlap, the leading group will not start maintaining the target overlap size with the following group until after a delay D equal to the sum of the time until the vehicle in the overlap has a chance to transmit a token for the leading group and the time until the token is committed. Before this delay, the overlap size can become smaller than 2L+2Δ. Therefore, the target overlap size has to be extended by e equal to the delay D times the speed of the trailing edge of the leading group in relation to the speed of the leading edge of the following group to ensure that the overlap size is always ≥ 2L+2Δ.

In summary, to handle the join delay and the delay until group members are informed about the overlap and start maintaining the target overlap size, vehicles are

allowed to use only the group that they are ≥ L + Δ from both group edges to transmit messages to their neighbors, and the target overlap size (O_T) is extended to 2L + 2Δ + e. 2L is the neighborhood span, 2Δ is the extra overlap to give vehicles time to join a new group, and e is the extra overlap to give a vehicle in the overlap time to inform other vehicles about the overlap and ensures that the overlap is always ≥ 2L + 2Δ. This value ensures that each vehicle in the network can always reach all of its neighbors in one group and is always accepted to a new group before it needs to use the new group to communicate with its neighbors.

3.5.4 Leaving groups

Since the group may move at a different speed than the individual members, vehicles may have to leave the group when the group span does not cover their positions anymore. However, we do not want vehicles near the edges of the group to rapidly leave and join the group because of small changes in the group edge positions. In this section, we show how to use hysteresis to prevent this situation. The positions for joining and leaving the group are slightly different.

A vehicle joins a new group when it crosses either edge of the new group and moves into the span of the group as described in section 3.5.1, but a vehicle does not leave a group of which it is a member until it is at a distance > the leave buffer behind the trailing edge of the group or at a distance > the leave buffer in front of the leading edge of the group. Figure 10 shows the points where vehicles join and leave a group. The trailing edge and leading edge of the group are at positions x_T and x_L, respectively. Vehicles moving forward in relation to the group join the group when they cross position

x_T. Vehicles moving backward in relation to the group join the group when they cross position x_L. Group members moving forward in relation to the group leave the group when they cross position x_L + leave buffer. Group members moving backward in relation to the group leave the group when they cross position x_T - leave buffer.

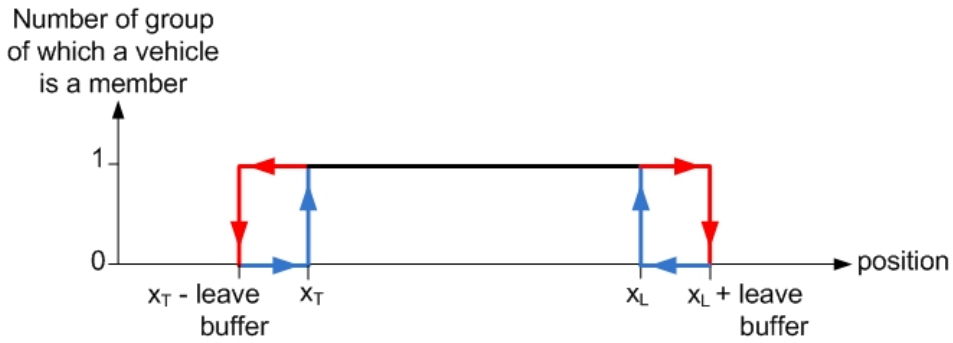

Figure 10. The points where vehicles join and leave a broadcast group

The leave buffer is set to v_{max}*max token commit delay, where v_{max} is the maximum vehicle speed and max token commit delay is the maximum delay until a token is committed, calculated from the maximum possible number of group members that can fit in a broadcast group. Because there is a token commit delay until a token that proposes new edge positions is committed, the edges of a group can stay unchanged for a period of time and then abruptly move to another position. The edge of a group can stay at position x for at most the max token commit delay interval and then abruptly move forward to position at most x + (v_{max}*max token commit delay). It is likely that vehicles that are outside the group span and at a distance > (v_{max}*max token commit delay) from the edge of the group will not move back into the group span. Therefore, the value v_{max}*max token commit delay is used for the leave buffer.

3.5.5 Splitting a group

We split a group when the group span is large enough to fit 2 groups that overlap with each other by the target overlap size (O_T) and each has a span ≥ the target group span (S_T) into one group. Note that for simplicity, we allow a group to take part in only one split or merge at a time. This section describes the split process in details.

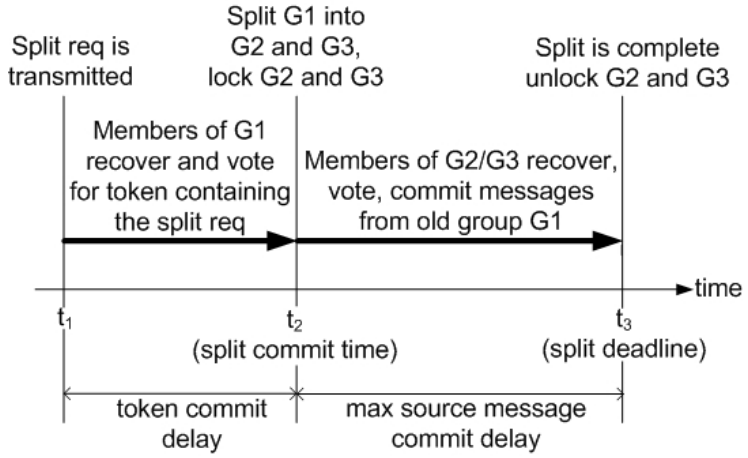

Figure 11. Timeline for splitting G1 into G2 and G3

Figure 11 shows the split process timeline for splitting a group G1 into two new groups G2 and G3. The split process contains the following steps:

1) A member of G1 proposes to split G1 by including a split request in its scheduled token for G1 at time t_1.

   A split is proposed if the span of G1 ≥ ($2*S_T$) - O_T and G1 is not locked. Each broadcast group has a lock so it can take part in only one split or merge at a time.

2) Members of G1 recover and vote for the token containing the split request as normal.

3) The members split G1 into two new groups G2 and G3 at the token commit time t_2 (also called the split commit time) if the token containing the split request is voted in and G1 is not currently locked.

The new groups G2 and G3 are locked until the split deadline t_3, which is the time that the split process will be complete. G2 and G3 have the same group span and overlap with each other by the target overlap size O_T as shown in Figure 12.

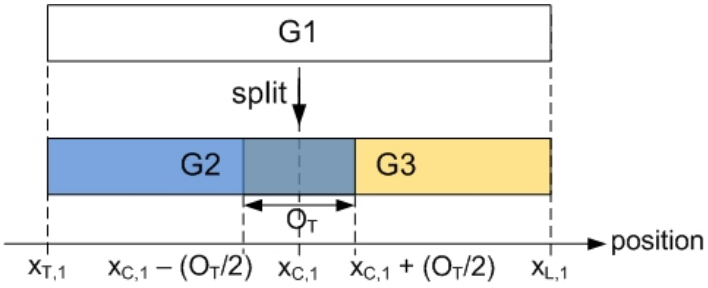

Figure 12. G1 splits into new groups G2 and G3

The member lists of G2 and G3 are created based on possible current positions of G1's members. Each member of G1 calculates the range of possible current positions of each member j of the group $[e_{j,min}, e_{j,max}]$. Then the record of member j is added to the member lists of all the new groups (G2/G3) that cover any position in the range $[e_{j,min}, e_{j,max}]$.

The calculations of $e_{j,min}$ and $e_{j,max}$ are based on the assumption that each vehicle can only move at a speed between 0 mph and a maximum speed v_{max} mph. The actual current position of member j can be anywhere in the range $[e_{j,min}, e_{j,max}]$. $e_{j,min}$ and $e_{j,max}$ are calculated as follows:

$$e_{j,min} = x_{j,r} \tag{6}$$

$$e_{j,max} = x_{j,r} + [v_{max} * (t - t_{j,r})] \tag{7}$$

where

$x_{j,r}$ is the most recently recorded position of member j.

$t_{j,r}$ is the time that member j was at position $x_{j,r}$.

t is the current time (split commit time t_2).

If the old group, G1, maintained the target overlap size (O_T) with some group behind it before it splits, the new group in the back, G2, continues to maintain the target overlap size with that group. If there is at least one member that has an entry in the member lists of both G2 and G3, then G3 maintains the target overlap size with G2.

4) All members of G2 and G3 continue recovering, voting, and committing messages from the old group G1. Messages that were sent before G1 split should not be dropped because of the split process.

5) After the split deadline t_3, there will be no more messages from the old group G1 that members of G2 and G3 have to recover, vote, and commit. The split process is considered complete, and the members of G2 and G3 can unlock these groups.

The split deadline t_3 is the max source message commit delay after the split commit time t_2. The max source message commit delay is the maximum delay until a source message is committed, which is calculated from the maximum possible number of group members that can fit in a broadcast group. All messages sent from the old group G1 will be committed by the max source message commit delay interval after t_2.

3.5.6 Merging groups

When one group overlaps the center of an adjacent group, we merge the two groups into a single group. This section describes how to merge two groups G1 and G2 into a new group G1′ in detail.

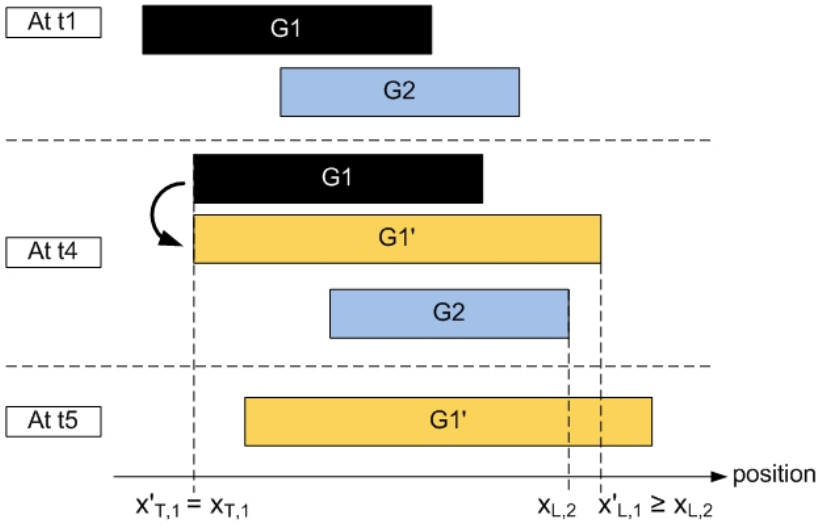

Figure 13. Merging G1 and G2 into a new group G1′

Our self-organizing protocol merges two groups G1 and G2 together by changing the group in the back, G1, to a new group G1′ that covers the span of both G1 and G2, then letting the members of G2 join the new group G1′and finally leave G2 as shown in Figure 13.

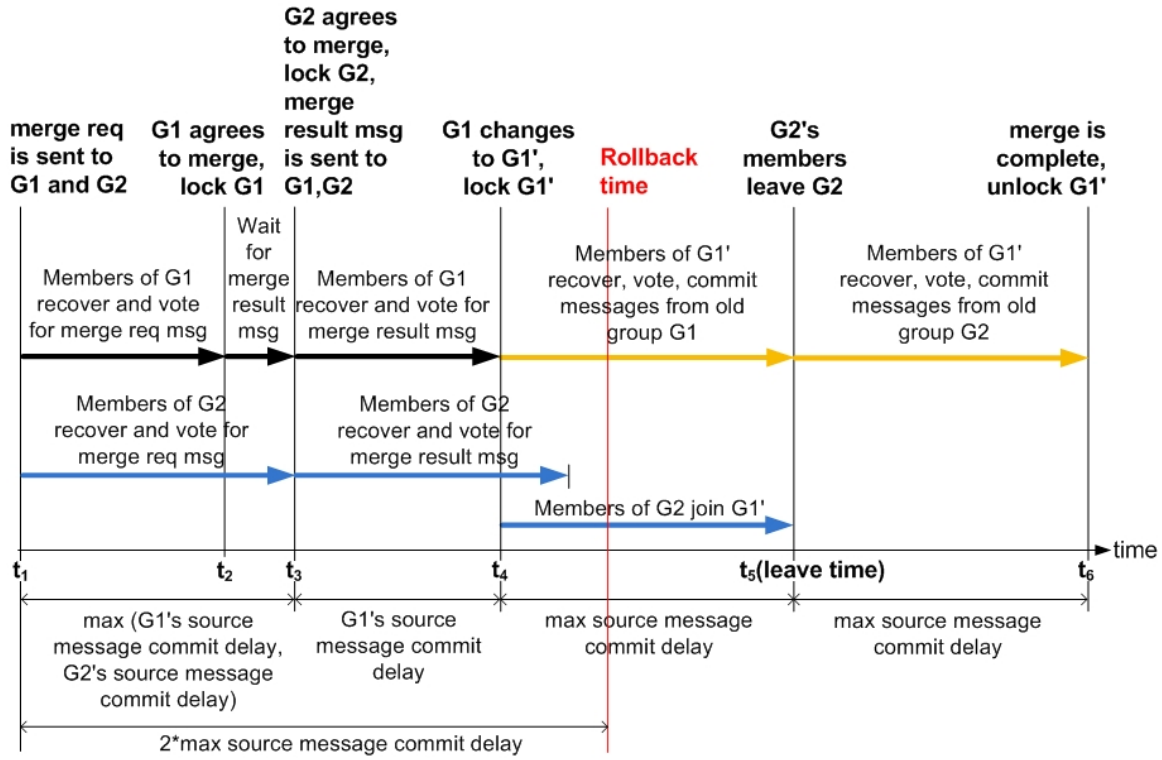

Figure 14. Timeline for merging G1 and G2 into a new group G1′

Figure 14 shows the timeline for merging G1 and G2 into G1′. The merge process contains the following steps:

1) A vehicle in the overlap between G1 and G2 proposes to merge G1 and G2 together if one of the groups overlaps the center of the other group and both groups are not locked.

   The vehicle proposes the merge by transmitting a merge request as a source message to both groups at time t_1. The proposing vehicle will be called the merge initiator.

2) Members of G1 and members of G2 recover, vote, and commit the merge request for their group as normal. At the commit time for the merge request (t2 for G1 and t3 for

G2), the members agree to merge if the merge request message is voted in and the group is not locked.

If the members agree to merge, they lock the group until the rollback time and wait for the merge result message from the merge initiator. If the rollback time has passed and no merge result message has been committed, then the members assume that the merge fails and unlock the group. The merge result message tells whether the other group with which they want to merge agrees to merge or not. The members can proceed to merge only if both groups agree to merge. Otherwise, the merge fails and the members unlock the group.

The rollback time is the time twice the max source message commit delay after the merge request is sent. By the first max source message commit delay interval after the merge request is sent, the merge request will have been committed at both G1 and G2, so the merge initiator will have known whether both groups agree to merge or not and sent a merge result message. By the second max source message commit delay interval, the merge result message will have been committed by both groups, so both groups will have known whether they can proceed to merge or not.

3) At t_3, the merge initiator knows whether both groups agree to merge or not, so it transmits a merge result message as a source message to both groups.

The merge initiator also includes the current position of the leading edge of G2 (x_2) in the message so that members of G1 can use this information to calculate the leading edge of the new group G1′ that covers the span of G2 in the next step.

4) Members of G1 and members of G2 recover, vote, and commit the merge result message for their group as normal. The members can proceed to merge only if the merge result message is voted in, and the message tells that both G1 and G2 agree to merge. Otherwise, the merge fails and the members unlock the group.

5) At the commit time of the merge result message at G1 (t_4), members of G1 proceed to merge by changing G1 to a new group G1′ that covers the span of both G1 and G2 and lock G1′. All members of G1 become members of G1′ and start passing tokens for G1′. The tokens also include the time t_4 that G1′ is created; which will be used by members of G2 to calculate the time that they can leave G2 in the next step.

   The trailing edge of G1′ is set to the same position as the trailing edge of G1 ($x'_{T,1} = x_{T,1}$) as shown in Figure 13; while the leading edge of G1′ is set to position $x'_{L,1}$ calculated from the leading edge position of G2 (x_2) included in the merge result message as follows:

$$x'_{L,1} = [x_2 + (v_{max}*\text{max token commit delay})] + [v_{max}*(t_4-t_3)] \quad (8)$$

   The calculation above is derived from the following ideas. Because there is a token commit delay until a token that proposes new edge positions is committed, the edges of a group can stay unchanged for a period of time and then abruptly move to another position. In our case, we know that the leading edge of G2 was at position x_2 at time t_3 when the merge result message was sent, but the edge may suddenly move to another position shortly after t_3. In the worst-case scenario, the edge can abruptly move from position x_2 to $x_2 + (v_{max}*\text{max token commit delay})$, as mentioned in section 3.5.4. Therefore, we assume that at time t_3, the leading edge of G2 was at

position x_2 + (v_{max}*max token commit delay), instead of just x_2, in the first part of Eq.(8). The second part of Eq.(8) is based on the fact that the speed of the leading edge of G2 cannot exceed the maximum vehicle speed v_{max}.

The trailing edge of G1′ moves normally, according to the method in section 3.5.2. If the old group G1 maintained the target overlap size O_T with some group behind it before it is changed to G1′, G1′ continues to maintain the target overlap size with that group.

The leading edge of G1′ does not move normally as the trailing edge; instead it moves with the maximum vehicle speed v_{max}. Since the speed of the leading edge of G2 cannot exceed v_{max}, moving the leading edge of G1′ in this manner ensures that G1′ always covers the span of G2 and all members of G2 will join G1′.

6) Members of G2 hear tokens from the new group G1′ and join G1′. Then they leave G2 at the leave time t_5. However, if the members of G2 commit the merge result message and the message reports that both groups agree to merge, but the members have not heard any token from the new group G1′ by the rollback time, they will assume that the merge failed and unlock G2.

The leave time t_5 is the max source message commit delay after the new group G1′ is created (t_4). All of G2's members will have successfully joined G1′ by the leave time, so they can use G1′ to communicate with their neighbors and do not need G2 anymore. The leave time can be calculated from the included time t_4 in the tokens from G1′.

7) All members of G1′ continue recovering, voting, and committing messages from the old groups G1 and G2.

8) After the merge deadline t_6, there will be no more messages from the old groups that members of G1′ have to recover, vote, or commit. The merge is considered complete, and all members of G1′ unlock the group and move both edges of the group normally according to the method in section 3.5.2.

   The merge deadline is double the max source message commit delay after G1′ is created (t_4). All messages from G1 will have been committed by the first max source message commit delay interval after t_4 and all messages from G2 will have been committed by the second max source message commit delay interval after t_4.

Chapter 4

Performance of the self-organizing protocol

In this chapter, we first list the metrics used to evaluate the performance of the self-organizing protocol. We then show how the protocol performs, and finally discuss the effects of message loss on the protocol.

4.1 Metrics

The following metrics are used for the performance evaluation.

1) Average number of times that a vehicle joins a new group per minute

2) Average number of groups of which a vehicle is a member

4.2 Average number of times that a vehicle joins a new group per minute

In this section, we calculate the upper bound of the average number of times that a vehicle joins a new group per minute (J_{max}). We confirm that the calculated J_{max} is correct by showing the results from the simulation of our self-organizing protocol. Then we compare the J_{max} of our self-organizing protocol with the J_{max} of stationary groups.

J_{max} is a function of the minimum group span and the speeds of vehicles in relation to the speeds of the edges of the groups. It is calculated as follows:

$$J_{max} = 60*\max(|v - v_G|)*0.44704/(2L) \quad (9)$$

where

v_G is the speeds of the edges of the groups in mph.

v is the vehicle speeds in mph.

2L is the neighborhood span in m.

The J_{max} of our self-organizing protocol is calculated as follows. Assume that vehicles in the network move with speeds between a minimum (v_{min}) and a maximum (v_{max}). Since the protocol moves broadcast groups with vehicles, v_G is between v_{min} and v_{max}. Therefore,

$$J_{max} = 60*(v_{max} - v_{min})*0.44704/(2L) \quad (10)$$

We have simulated the self-organizing protocol and confirmed that the average number of times a vehicle joins a new group is below the calculated upper bound (J_{max}). 33 vehicles are put in 3 broadcast groups where adjacent groups overlap by the target overlap size of 252.74 m. Each group has the same target group span of 625 m. Each vehicle has the same neighborhood span (2L) of 250 m and is set 40 meters apart from the adjacent vehicles. The vehicle randomly chooses a speed between v_{min} of 60 mph and v_{max} of 80 mph (the mean speed is 70 mph) and moves with the chosen speed during the entire simulation. The results show that the average number of times that a vehicle joins a new group (J) is 0.58 times/min; which is below J_{max} of 2.15 times/min from the calculation. The J from the simulation is shown as the red point at mean vehicle speed 70 mph in Figure 15.

The J_{max} of stationary groups is calculated by $v_G = 0$. So,

$$J_{max} = 60*v_{max}*0.44704/(2L) \quad (11)$$

Figure 15 compares the J_{max} of our self-organizing protocol with the J_{max} of stationary groups at different mean vehicle speeds. The neighborhood span (2L) is assumed to be 250 m. The mean vehicle speed is varied between 0 to 70 mph. The minimum vehicle speed (v_{min}) and the maximum vehicle speed (v_{max}) are 14.286 % below and 14.286 % above the mean vehicle speed, respectively.

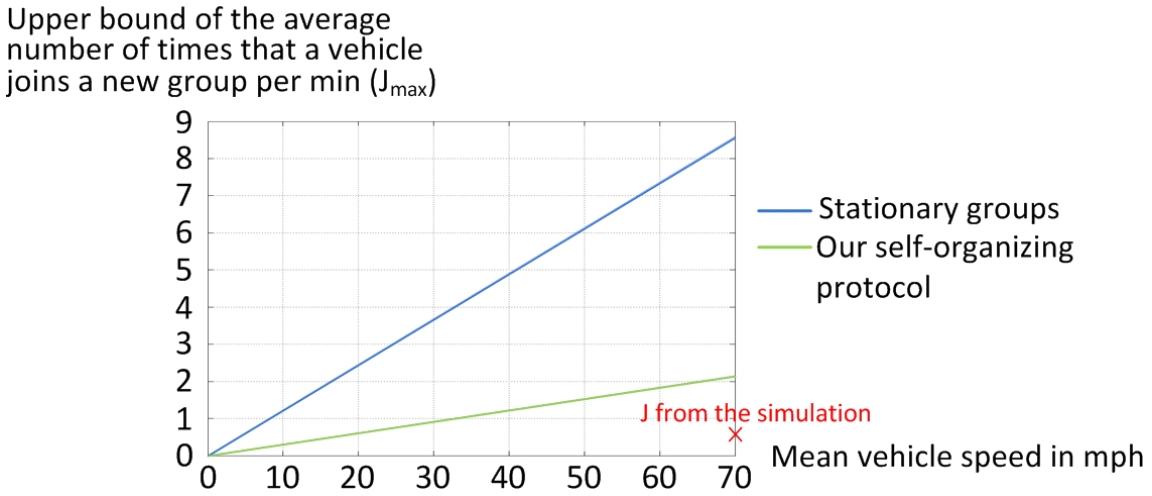

Figure 15. Upper bound of the average number of times that a vehicle joins a new group per minute at different mean vehicle speeds

Figure 15 shows that the J_{max} of stationary groups is much higher than the J_{max} of our self-organizing protocol. For example at the mean speed 70 mph (v_{min} is 60 mph and v_{max} is 80 mph), J_{max} of stationary groups is 8.58 times/min, while J_{max} of our self-organizing protocol is. 2.15 times/min. Therefore, our self-organizing protocol results in fewer join request messages and recovery messages for the join requests than stationary groups.

4.3 Average number of groups of which a vehicle is a member

In this section, we calculate the upper bound and lower bound of the average number of groups of which a vehicle is a member (N). Then we confirm that the calculated bounds are correct by showing the results from the simulation.

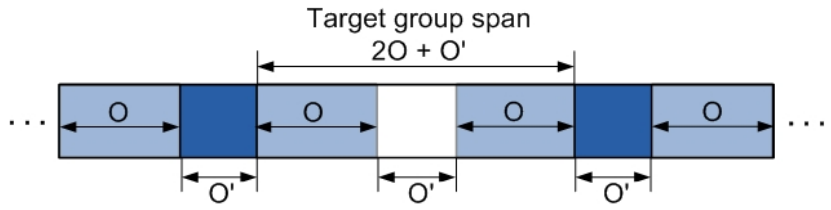

Figure 16. Overlapping groups that have the same target group span and overlap by O

Figure 16 is used for the calculations. O and O' are the sizes of the overlapping and non-overlapping parts of the group, respectively. Each group has the same target group span of 2O+O', and every pair of adjacent groups overlap by O.

N depends on the portion of the overlapping and non-overlapping parts of a group. According to Figure 16,

$$\mathrm{N} = \lim_{n\to\infty} \frac{[2(n+1)\mathrm{O}] + (n\mathrm{O}')}{[(n+1)\mathrm{O}] + (n\mathrm{O}')} = 2 - \left(\frac{\mathrm{O}'}{\mathrm{O}+\mathrm{O}'}\right) \tag{12}$$

The upper bound of N (N_{max}) is calculated from the smallest value of O' (O' = 0). So, N_{max} = 2. The lower bound of N (N_{min}) is calculated from O = 2L and the largest value of O' (O' = 500-3L). O' is (500-3L) because it is the largest value that ensures a group will split into smaller groups before its span exceeds 1000 m. As a result, group members can use dedicated short-range communications (DSRC) [44], which support the

maximum transmission range of 1000 m, to send messages/tokens to all other group members without the need for message forwarding. Therefore,

$$N_{min} = 2- [(500-3L)/(500-L)] \quad (13)$$

We have simulated our self-organizing protocol and confirmed that the average number of groups of which a vehicle is a member is between the calculated lower bound N_{min} and upper bound N_{max}. The same parameters described in section 4.2 are used for the simulation. The neighborhood span (2L) is 250 m. The target group span is 625 m. O is equal to the target overlap size of 252.74 m and O' is 119.52 m. The results show that the average number of groups of which a vehicle is a member is 1.7, which is between the calculated lower bound of 1.67 and the upper bound of 2.

4.4 Effects of message loss

The self-organizing protocol works correctly in an environment with message loss. The same parameters described in sections 4.2 and 4.3 and the following four message loss models are used for the simulation.

1) Vehicles independently lose a message with a probability of 0.01.
2) Vehicles independently lose a message with a probability of 0.001.
3) Message loss at all vehicles is fully correlated. All vehicles lose a message with a probability of 0.01.
4) Message loss at all vehicles is fully correlated. All vehicles lose a message with a probability of 0.001

The results show that all 33 vehicles were always members of at least one broadcast group that contains themselves and all of their neighbors at all times.

Chapter 5

Providing the RNP guarantees

As mentioned in chapter 2, RNP is intended to provide guaranteed message delivery from each vehicle in a VANET to all of its neighbors within a bounded delay, ensure that all the neighbors that receive the same messages sequence them in the same order and use each of them at the same time, and provide the neighbors the knowledge of whether all of the other neighbors have received the message or which neighbors are missing the message. This chapter describes the mechanism that provides these guarantees, which is the second part of RNP. The mechanism transfers a set of guarantees provided by the underlying protocol, the modified version of M-RBP, from the broadcast group level to the neighborhood level. In this mechanism, a vehicle transmits application-level messages to all its neighbors in one broadcast group that it is $\geq L + \Delta$ from both edges of the group. Only the neighbors of the vehicle recover, vote, and commit the messages. Note that M-RBP tokens and messages related to the self-organizing protocol, i.e. join request messages, merge request messages, and merge result messages, are still recovered, voted, and committed by all members of the group as normal.

The timeline for transmitting, voting, and committing application-level messages in RNP is the same as the timeline for M-RBP; however, we repeat it here in Figure 17 for convenience and to make it easier to understand the mechanism described in this chapter.

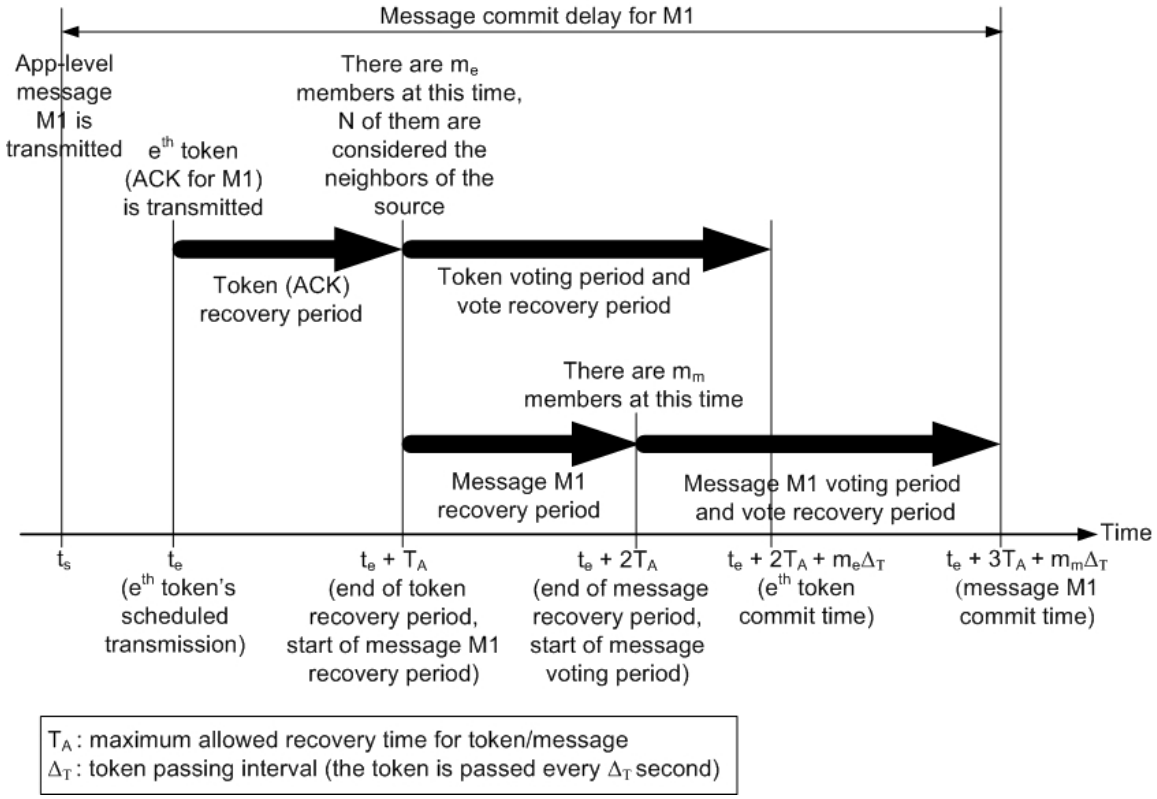

Figure 17. Timeline for using RNP to transmit an application-level message

To allow only the neighbors of the source to recover, vote, and commit the application-level message, the source includes its current position (x_s), current speed (v_s), current acceleration rate (a_s), and the current time (t_s) in the message. This information is transferred to the token that acknowledges the message (e^{th} token in Figure 17), which is then recovered, voted, and committed by all group members. At the start of the message recovery period ($t_e + T_A$), all group members use this information to determine whether or not to recover, vote, and commit the message by estimating the current position of the source to determine the total number of the source's neighbors and whether they are among these neighbors.

We will now provide the details on how group members use the information from the token to determine the total number of the source's neighbors. At the time $t_e + T_A$, all members that have received the token use the information included in the token to estimate the current position of the source (e_s) and calculate the range of possible current positions of each member j of the group [$e_{j,min}$, $e_{j,max}$]. Only member j whose possible current positions are within the distance L (half the neighborhood span) from the estimated current position of the source is considered a neighbor of the source and has to recover, vote, and commit the message. In other words, member j is considered a neighbor of the source if:

$$e_s - L \le e_{j,min} \le e_s + L \quad \text{or} \quad e_s - L \le e_{j,max} \le e_s + L$$

The current position of the source (e_s) is calculated as follows:

$$e_s = x_s + [v_s*(t_e + T_A - t_s)] + 0.5*a_s*(t_e + T_A - t_s)^2 \tag{14}$$

The range of possible current positions of each member j [$e_{j,min}$, $e_{j,max}$] is calculated based on the assumption that each vehicle can only move at a speed between 0 and a maximum speed v_{max}. The actual current position of member j can be anywhere between $e_{j,min}$ and $e_{j,max}$. The calculations of $e_{j,min}$ and $e_{j,max}$ are as follows:

$$e_{j,min} = x_{j,r} \tag{15}$$

$$e_{j,max} = x_{j,r} + [v_{max}*(t_e + T_A - t_{j,r})] \tag{16}$$

where

$x_{j,r}$ is the most recently recorded position of vehicle j

$t_{j,r}$ is the time that vehicle j was at position $x_{j,r}$

v_{max} is the maximum vehicle speed

The voting procedure for the application-level message is the same as the voting procedure of M-RBP, except that only the neighbors of the source, rather than all group members, vote for the message. The expected number of votes is equal to the number of neighbors, rather than the number of group members. At the end of the message recovery period ($t_e + 2T_A$), each neighbor includes its vote for the message in its scheduled token. A "YES" vote is included if the neighbor has received the application-level message and a "NO" vote is included otherwise.

The message can be committed only at the message commit time ($t_e + 3T_A + m_m\Delta_T$). Suppose at the start of the message recovery period ($t_e + T_A$), N members are considered the neighbors of the source. A_i and B_i are the total number of YES and NO votes respectively that a neighbor i has received. At the message commit time, each neighbor i decides whether or not to commit the message by using the following rules:

- If $A_i > N/2$, then the actual number of YES votes transmitted is > N/2, so neighbor i commits the message.
- If $B_i \geq N/2$, then the actual number of NO votes transmitted is ≥ N/2 and the actual number of YES votes transmitted is ≤ N/2, so neighbor i does not commit the message.
- If $B_i < N/2$, and $A_i \leq N/2$, neighbor i is uncertain whether or not the actual number of YES votes transmitted is ≤ N/2, so it leaves the group and might also take some application-specific preventive action e.g., increasing the following distance from the preceding vehicle.
- If $A_i > N/2$, but neighbor i has not recovered the message, then it leaves the group

and might also take some application-specific preventive action. This is the case when the other neighbors decide to commit the message but neighbor i does not have the message.

(Note that depending on the VANET application, the rules for deciding whether to commit the message can be either majority vote, as described above, or unanimous vote.)

To ensure that all the neighbors that receive the same message sequence the message in the same order, vehicles order received messages based on the message commit times. When two vehicles receive two messages transmitted in two different broadcast groups, they place the messages in the same order since each message is only transmitted in one broadcast group.

At the message commit time ($t_e + 3T_A + m_m\Delta_T$), a neighbor can know whether all of the other neighbors have received the message, or which neighbors are missing it by looking at the votes it has received from other neighbors. All neighbors from which it has received YES votes have successfully received the message, while all neighbors from which it has received NO votes missed the message. Neighbors it has not received votes from may have missed the message.

Chapter 6

Performance of RNP

This chapter shows the performance of RNP in terms of the maximum the delay until RNP guarantees that all neighbors of a source successfully receive and commit an application-level message from the source, and the average number of messages that occur when an application-level message is transmitted. We first describe the message loss model that we use for the performance evaluation in section 6.2. In section 6.3, we then describe how to calculate the maximum delay until all neighbors successfully receive and commit an application-level message and show that this delay is only 0.081 seconds in the case that the neighborhood span (2L) is 250 meters, the target group span is 625 meters, the M-RBP token is passed every 0.001 second, and the maximum number of token/message recovery iterations allowed is 4.

There are two types of messages that occur when RNP is used. The first is the messages from the M-RBP token passing and token recovery mechanism. The second is the messages from application-level message transmission. In section 6.4, we describe the calculations of both types of messages from RNP but we only compare the second type of messages i.e., the average number of messages occurring from one application-level message transmission from RNP with a simple ARQ protocol that allows a source to repeatedly transmit an application-level message until it receives ACKs from all of its neighbors. We show that RNP results in only about 1 message per one application-level message transmission on average, while the simple ARQ protocol results in about N+1

messages. We will not compare the first type of messages between the two protocols. Since this type of messages from RNP allows source vehicles to know who their neighbors are, the simple ARQ protocol also needs to have some mechanism e.g., periodic keep-alive messages that report vehicle positions, to provide the same information to the sources.

6.1 Metrics

The following metrics are used for the performance evaluation of RNP.

1) Maximum delay until all neighbors successfully receive and commit an application-level message (D_{max})

2) Average number of messages occurring from one application-level message transmission

6.2 Message loss model and correlation of message loss

We model message loss and correlation of message loss as follows. When a message is transmitted, the channel can be either good or bad. The channel is bad with a probability P_B and good with a probability 1- P_B. When the channel is good, all receiving vehicles receive the message with a probability 1. When the channel is bad, each receiving vehicle loses the message with a probability $P_{L|B}$ and receives the message with a probability 1- $P_{L|B}$.

From this concept, probability that a receiving vehicle loses a message (P_L) is

$$P_L = P_B * P_{L|B} \quad (17)$$

Correlation of message loss at 2 receiving vehicles (ρ) can be calculated as:

$$\rho = \frac{P_{L|B} - P_L}{1 - P_L} \quad (18)$$

ρ is 0 in the case that receiving vehicles independently lose a message and 1 in the case that message loss at receiving vehicles is fully correlated. When $\rho = 0$, the channel when a message is transmitted is always bad ($P_B = 1$) and each receiving vehicle loses the message with probability P_L ($P_{L|B} = P_L$). On the other hand, when $\rho = 1$, the channel when a message is transmitted is either good or bad and is bad with probability P_L ($P_B = P_L$). All receiving vehicles receive the message if the channel is good and all of them lose the message if the channel is bad ($P_{L|B} = 1$).

6.3 Maximum delay until all neighbors successfully receive and commit an application-level message (D_{max})

In this section, we describe how to calculate the upper bound of the delay or maximum delay until RNP guarantees that all neighbors of a source successfully receive and commit an application-level message sent from the source (D_{max}). We also describe the calculation of the probability that not all neighbors receive and commit an application-level message within D_{max} in an environment with message loss, and show an example of D_{max}. The idea is to determine D_{max} such that in an environment with message

loss, the probability that not all neighbors receive and commit an application-level message within D_{max} is $\leq 10^{-6}$.

We will first give the equation to calculate D_{max}, which is the maximum interval from the time that the source transmits an application-level message, to the time that the message is committed. This interval includes the duration until the first token that acknowledges the message is transmitted and the token/message recovery period. The details on how to calculate these two intervals are described in section 6.3.1. Then, in section 6.3.2, we describe the calculation of the probability that not all neighbors receive and commit an application-level message within D_{max} in an environment in which a vehicle loses a message with probability P_L. Finally, in section 6.3.3, we show an example of D_{max} in the case that the neighborhood span (2L) is 250 meters, the target group span is 625 meters, the M-RBP token is passed every 0.001 second, and vehicles move at speed 100 km/h (62.14 mph). We show that D_{max} in this case is only 0.081 seconds and if the probability that a vehicle loses a message (P_L) is ≤ 0.01, then the probability that not all neighbors receive and commit an application-level message within 0.081 second is $\leq 0.815*10^{-7}$.

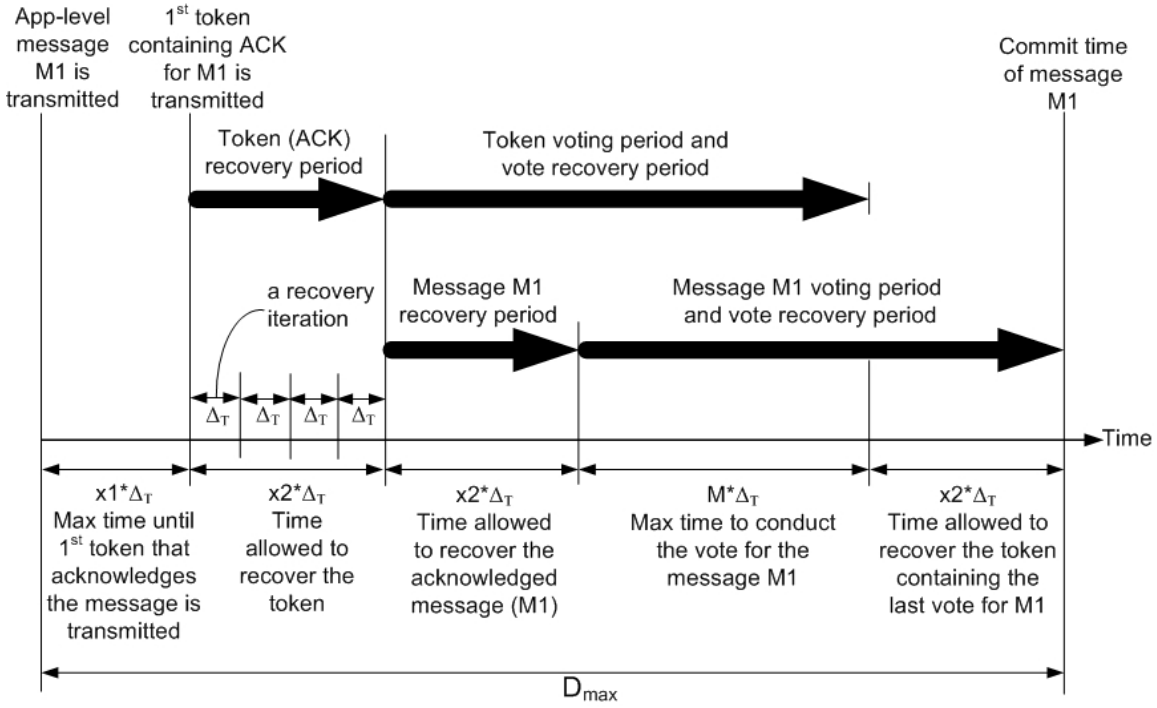

Figure 18. Timeline for using RNP marked with variables related to D_{max} calculation

Figure 18 shows the timeline for using RNP to transmit an application-level message. This timeline is the same as Figure 17 in chapter 5; we repeat it here and mark it with the variables related to D_{max} calculation to make the calculation easier to understand. D_{max} is the maximum interval from the time that the source transmits an application-level message to the time that the message is committed.

$$D_{max} = [x1 + 3(x2) + M] * \Delta_T \quad (19)$$

where

Δ_T is the token passing interval (the token is passed every Δ_T seconds).

x1 is the maximum number of times that the source has to transmit the application-level message until the first token that acknowledges the message is transmitted.

x2 is the maximum number of token/message recovery iterations allowed.

M is the maximum number of group members.

$x1*\Delta_T$ is the maximum time until the first token acknowledging the application-level message is transmitted. Because the token is passed every Δ_T seconds and the current token site acknowledges all messages received during the period Δ_T seconds before its scheduled token transmission time, a message source transmits its application-level message and waits for Δ_T seconds for an ACK from the current token site before retransmitting the message. To calculate D_{max}, we will use x1, which makes the probability that no ACKs are sent to acknowledge the application-level message after the source transmits the message x1 times $\leq 10^{-6}$. The calculation of x1 will be described in section 6.3.1.

In RNP, we fix the length of the token/message recovery period to $x2*\Delta_T$. All group members that miss the scheduled token/message have x2 recovery iterations, each of length Δ_T, to recover the token/message. We will use x2, which makes the probability that not all group members receive the token/message by the end of the x2 recovery iterations $\leq 10^{-6}$. The calculation of x2 will be described in section 6.3.1.

$(M+x2)*\Delta_T$ is the maximum length of the message voting and vote recovery period. Because a vehicle votes for the message by including its vote in its scheduled token, the maximum length of the message voting period is calculated from the maximum number of group members, M.

Since we want to calculate the upper bound of the delay, we assume that $\rho = 0$ i.e., vehicles independently lose a message. We also assume that the probability that a vehicle loses a message (P_L) is ≤ 0.01.

6.3.1 How to determine x1 and x2 for calculating D_{max}

In this section, we show how to calculate maximum number of times that the source has to transmit the application-level message until the first token acknowledging the message is transmitted, x1, and the maximum number of token/message recovery iterations allowed, x2.

How to determine x1

The probability that no ACKs are sent to acknowledge the application-level message after the source transmits the message x1 times is the probability that none of the x1 token sites receive the message, which is equal to $(P_L)^{x1}$. From the assumption that $P_L \leq 0.01$, we use x1 = 3 for the calculation of D_{max} because the probability that no ACKs are sent after the source transmits the message three times $\leq 10^{-6}$.

How to determine x2

During each token recovery iteration, each member that misses the scheduled token sends a negative acknowledgement (NACK) to the token site to request the retransmission of the token. The token site retransmits the token only if it receives at least one NACK from the group members. Therefore, a member successfully recovers the token in the iteration if both 1) the token site retransmits the token in that iteration; and 2) the member receives the retransmitted token.

If the number of recovery iterations that a member needs to successfully recover the token is x2 when it is the only member that misses the token, then the number of recovery iterations that a member needs to successfully recover the token is < x2 when

there is more than one member that misses the token. Therefore, assuming that the maximum number of group members, M, is 70 and $P_L \leq 0.01$, we choose x2 = 4 because it makes the probability that not all M-1 members (other than the token site) receive the token by the end of x2 recovery iterations, $1-\{ 1 - [P_L (P_L + (1- P_L)(P_L))^{x2}] \}^{M-1}$, smaller than or equal to $1.08*10^{-7}$; which is small enough.

We allow four recovery iterations for the application-level message recovery period as well. Since only a subset of the group members who are the neighbors of the source needs to recover for the application-level message, four recovery iterations are sufficient for all neighbors to recover the application-level message.

6.3.2 Probability that not all neighbors commit an application-level message within the maximum delay D_{max}

In this section, we calculate the upper bound of the probability that not all neighbors commit an application-level message within D_{max} when we allow the maximum four recovery iterations for both token recovery and application-level message recovery. To calculate the upper bound of the probability, we will assume the worst case in that there are M members in the group, N of them are neighbors of the source that transmits an application-level message, the token site that transmits the token containing the ACK for the application-level message is not one of the N neighbors, and $N < \lfloor M/2 \rfloor$. The last assumption requires other group members in addition to the N neighbors and the token site that transmits the token to receive the token to make the total number of YES votes for the token > M/2, which is the condition that allows the token to be committed.

The upper bound of the probability that not all neighbors commit an application-level message within D_{max} is calculated as follows:

P(not all N neighbors commit an application-level message within D_{max})

= 1- P(all N neighbors commit an application-level message within D_{max})

A neighbor commits an application-level message when it commits both the token containing the ACK for the message and the message. Therefore,

P(all N neighbors commit an application-level message within D_{max})

= P(all N neighbors commit the token containing the ACK for the application-level message)
* P(all N neighbors receive the application-level message)
* P(all N neighbors receive > N/2 YES votes for the application-level message)

P(all N neighbors commit the token containing the ACK for the application-level message)

= P(> M/2 group members receive the token, all N neighbors receive the token, all N neighbors receive > M/2 YES votes for the token)

=

$$\sum_{i=\lfloor M/2 \rfloor - N}^{M-N-1} \left\{ \binom{M-N-1}{i} (P_R)^{N+i} (1-P_R)^{M-N-i-1} \left[\sum_{j=\lfloor M/2 \rfloor + 1}^{N+i+1} \binom{N+i+1}{j} [P_R]^j [1-P_R]^{N+i+1-j} \right]^N \right\} \quad (20)$$

where

P_R = P(a vehicle receives a token/message by the end of 4 token/message recovery iterations)

$$= 1 - [\, P_L \, (P_L + (1- P_L)(P_L))^4 \,] \quad (21)$$

P(all N neighbors receive the application-level message) $= [P_R]^N$ (22)

P(all N neighbors receive > N/2 YES votes for the application-level message)

$$= \left\{ \sum_{i=\lfloor N/2 \rfloor+1}^{N} \binom{N}{i} (P_R)^i (1-P_R)^{N-i} \right\}^N \quad (23)$$

Therefore, P(not all N neighbors commit an application-level message within D_{max}) is 1- [(Eq.20)* (Eq.22)* (Eq.23)].

6.3.3 An example value of the maximum delay D_{max}

This section provides an example of the maximum delay D_{max} and the probability that not all neighbors commit an application-level message within D_{max} for the following conditions:

- The neighborhood span (2L) is 250 meters.
- The target group span is 625 meters.
- Consider only one lane of a highway.
- All vehicles move with the same speed (v) of 100 km/h (62.14 mph).
- Every vehicle has a communication device and uses RNP to communicate with its neighbors.
- Every vehicle has the same vehicle length (*l*) of 4.3 meters [45].
- Each vehicle maintains a safe following distance (H) of $v*(D_{max} + D_B)/3.6$ meters from the preceding vehicle. D_B is the mechanical delay until the brake is applied, which is 0.1 second [46].
- x1 = 3, x2 = 4.
- Token passing interval Δ_T = 0.001 second.

We first determine the maximum number of group members, M, in order to use it to calculate the delay D_{max}. M can be calculated from solving the following three equations:

$$H = 100*(D_{max} + 0.1)/3.6 \quad (24)$$

$$D_{max} = (15 + M)*0.001 \quad (25)$$

$$M = 625/(H+4.3) \quad (26)$$

From solving equations (Eq.24)-(Eq.26), M is equal to 66.8. So there can be at most, 66 members in the group and at most, 26 neighbors in the neighborhood span. Therefore, the maximum delay D_{max} in this case is 0.081 second. If the probability that a vehicle loses a message (P_L) is ≤ 0.01, then the probability that not all 26 neighbors commit an application-level message within 0.081 second is $\leq 0.815*10^{-7}$, which is very small.

6.4 Average number of messages occurring from one application-level message transmission

In this section, we describe how to calculate the average number of messages occurring from one application-level message transmission when RNP is used and when a simple ARQ protocol that allows a source to repeatedly transmit an application-level message until it receives ACKs from all of its neighbors is used. We verify the results from the calculations with the results from simulations and compare the average number of messages that occur from one application-level message transmission between the two

protocols. The calculation of the average number of messages for the simple ARQ protocol is described in section 6.4.1 and the calculation for RNP is described in section 6.4.2. As mentioned previously, there are two types of messages that occur when RNP is used: the messages from the M-RBP token passing and token recovery mechanism and the messages from application-level message transmission. In section 6.4.2, we show how to calculate both types of messages; however, in section 6.4.3, we only compare the average number of messages from application-level message transmission between the two protocols. In section 6.4.3, we show that the average number of messages from RNP is much smaller than the one from the simple ARQ protocol. For example, in the ideal case when there is no message loss, RNP results in the average number of messages of 1 per one application-level message transmission, independent of the number of neighbors (N) of the source, while the simple ARQ protocol results in the average number of messages of N+1. When N = 30 and vehicles independently lose a message with probability 0.01, RNP results in the average number of messages of only 1.5889, while the simple ARQ protocol results in the average number of messages of 31.7679.

6.4.1 Using a simple ARQ protocol to transmit messages

In this section, we first specify the rules of the simple ARQ protocol in section 6.4.1.1. Then, based on these rules, we describe how to calculate the average number of messages that occur from using the protocol to transmit an application-level message. This average number of messages is the sum of the average number of application-level messages transmitted from the source until it receives an ACK from each of the

neighbors described in section 6.4.1.2 and the average number of ACKs from all neighbors to the source described in section 6.4.1.3.

6.4.1.1 Rules of the simple ARQ protocol

We assume that the simple ARQ protocol uses the following rules to transmit and acknowledge an application-level message. The message source transmits the message until it receives an ACK from each of its neighbors. It transmits the message by broadcasting the message only to the neighbors from which it has not received an ACK. Each neighbor sends an ACK to the source every time it receives the message from the source.

6.4.1.2 Average number of application-level messages from the source to all neighbors

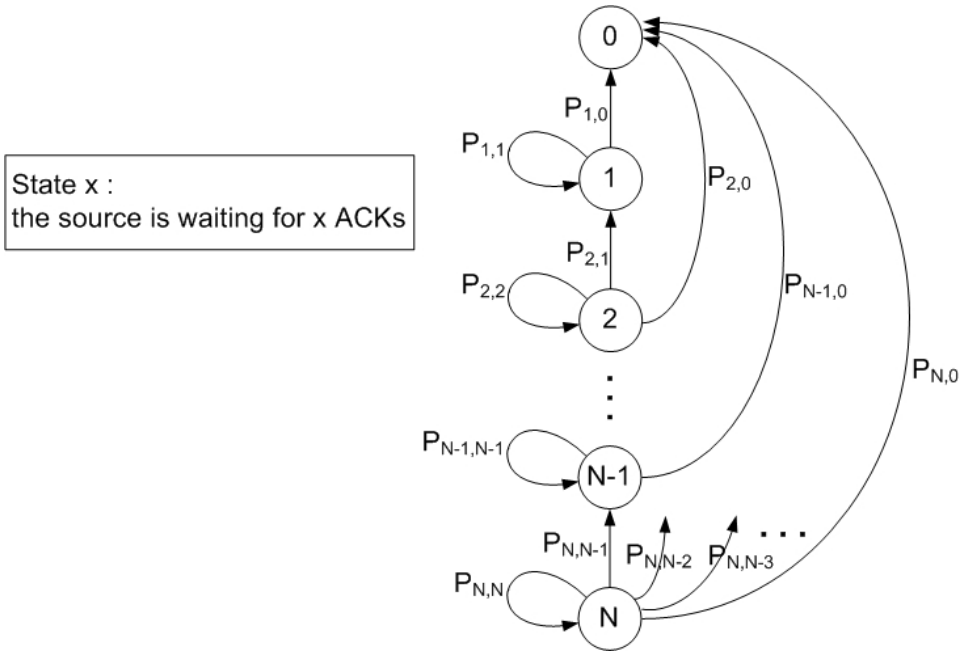

Figure 19. State transition diagram for calculating the average number of application-level messages transmitted from the source to N neighbors

Figure 19 shows the state transition diagram used for calculating the average number of application-level messages transmitted from the source until it receives an ACK from each of its N neighbors, where N is the total number of the source's neighbors. State x means the source is waiting for x ACKs from x neighbors. The average number of application-level messages that we want is the average number of steps needed to move from state N to state 0 or the expected number of steps before absorption if starting from state N.

The expected number of steps before absorption if starting from state N can be calculated from transition probabilities $P_{n,m}$ from every possible state n to every possible state m. Then the transition matrix **P** can be constructed from $P_{n,m}$ and matrix **P** can be used to find the fundamental matrix **N**. Finally, the expected number of steps before absorption if starting from state N, or the average number of application-level messages transmitted from the source until it receives an ACK from all N neighbors, is the sum over the Nth row of the fundamental matrix **N**.

We now show how to calculate transition probabilities $P_{n,m}$. The source moves from state n to m when it successfully receives exactly (n-m) ACKs from (n-m) neighbors. This event happens when ≥ (n-m) neighbors receive the message from the source, so the number of transmitted ACKs ≥ (n-m). The calculation of $P_{n,m}$ is as follows:

For m ≤ n,

$$P_{n.m} = \left[\binom{n}{n-m}(1-P_B)\left(1-P_B P_{L|B}\right)^{n-m}\left(P_B P_{L|B}\right)^{m}\right] + \left[\sum_{i=n-m}^{n}\binom{n}{i}\binom{i}{n-m}P_B\left(1-P_{L|B}\right)^{i}\left(P_{L|B}\right)^{n-i}\left(1-P_B P_{L|B}\right)^{n-m}\left(P_B P_{L|B}\right)^{i-(n-m)}\right] \quad (27)$$

For $m > n$, $P_{n,m} = 0$.

As an example, if the source has 30 neighbors ($N = 30$) and vehicles independently lose a message ($\rho = 0$) with a probability $P_L = 0.01$, then the average number of application-level messages transmitted from the source until it receives an ACK from all 30 neighbors from the calculation is 1.4649.

We have verified the calculation result by simulations. The result from the simulation is 1.4649, which is the same as the calculation.

6.4.1.3 Average number of ACKs from all neighbors to the source

Since the source independently loses ACKs from different neighbors, the average number of ACKs transmitted from all neighbors = $N*n_A$, where n_A is the average number of ACKs that a neighbor transmits until the source successfully receives it.

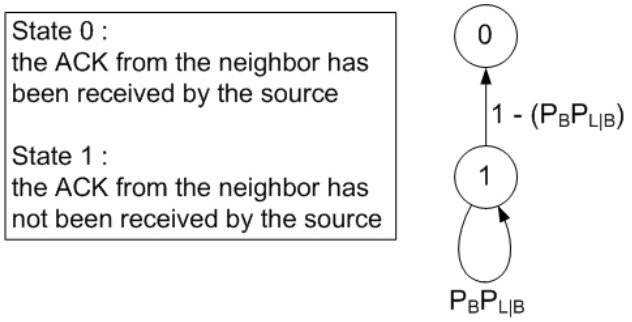

Figure 20. State transition diagram for calculating the average number of ACKs from a neighbor to the source (n_A)

Figure 20 shows the state transition diagram used for calculating n_A. A neighbor can stop transmitting ACK when the source successfully receives its ACK and hence stops sending the application-level message to it. n_A can be calculated as:

$$n_A = 1/[1-(P_B P_{L|B})] \quad (28)$$

As an example, if the source has 30 neighbors (N = 30) and vehicles independently lose a message ($\rho = 0$) with a probability $P_L = 0.01$, the average number of ACKs transmitted from all 30 neighbors to the source from the calculation is 30.303. This is close to the value of 30.3037 from the simulation.

6.4.2 Using RNP to transmit messages

Two types of messages that occur from using RNP: messages from the M-RBP token passing and token recovery mechanism and messages from application-level message transmissions will be described in section 6.4.2.1. Then we show how to calculate both types of messages in sections 6.4.2.2-6.4.2.6. Sections 6.4.2.2-6.4.2.4 describe the calculations of the average numbers of messages occurring from an application-level message transmission, while section 6.4.2.5 and 6.4.2.6 describe the calculations of the average numbers of tokens/messages occurring from the M-RBP token passing and token recovery mechanism.

6.4.2.1 Messages from RNP

There are two types of messages that occur when RNP is used.

1) The messages from the M-RBP token passing and token recovery mechanism.

   This type of message occurs even when no application-level messages are transmitted. It provides the source vehicles the information of who their neighbors are i.e., who are expected to receive messages from the sources. This type of messages consists of:

- scheduled tokens transmitted from the token sites of the group. These tokens also carry ACKs for received application-level messages.
- NACKs transmitted from group members that miss scheduled tokens to the token sites to request a token retransmission.
- retransmitted tokens from the token sites in response to the NACKs from the group members that missed the tokens.

2) The messages from application-level message transmissions

When an application-level message is transmitted, there are additional messages in addition to the first type of messages described above. The additional messages consist of:

- an original application-level message and retransmitted messages from the source until it receives an ACK for the message.
- NACKs transmitted from the neighbors of the source that miss the application-level message to the token site that sends the ACK for the message to request a message retransmission.
- Retransmitted application-level messages from the token site in response to the NACKs from the neighbors that miss the message.

6.4.2.2 Average number of application-level messages transmitted from the source until receiving an ACK

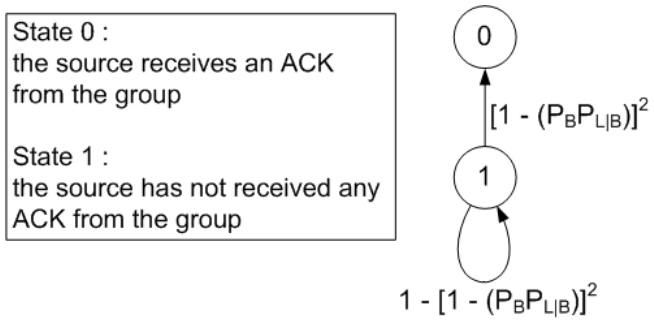

Figure 21. State transition diagram for calculating the average number of application-level messages transmitted from the source until receiving an ACK

As shown in Figure 21, the source receives an ACK from the group and can stop transmitting the message when both the current token site of the group receives the message from the source and transmits an ACK, and the source receives the ACK from the token site. Therefore,

Average number of application-level messages transmitted from the source

$$= 1/[(1 - P_B P_{L|B})^2] \quad (29)$$

As an example, suppose that vehicles independently lose a message ($\rho = 0$) with a probability $P_L = 0.01$. The average number of application-level messages transmitted from the source until it receives an ACK from the group from the calculation is 1.0203, which is close to the value 1.0205 from the simulation.

6.4.2.3 Average number of application-level messages retransmitted from the token site to neighbors that miss the message

After the source receives an ACK from the group and stops transmitting the application-level message, the number of neighbors that still have not received the message (i) can be any value between 0 and N, where N is the total number of neighbors of the source. The average number of application-level messages retransmitted from the token site that acknowledges the message to the neighbors that miss the message is calculated as follows:

Average number of application-level messages retransmitted from the token site

$$= \sum_{i=0}^{N} P_i S_i \tag{30}$$

where

P_i is the probability that there are i neighbors that miss the message transmitted from the source.

S_i is the average number of application-level messages that the token site has to retransmit until all i neighbors that have missed the message from the source receive the message. $S_i = 0$ when $i = 0$, and $S_i > 0$ otherwise.

P_i is calculated from the worst-case assumption that the source only transmits the message once, after which it receives an ACK from the group and stops transmitting the message. The calculation of P_i is as follows:

The case that i = 0 occurs when none of the N neighbors miss the message from the source given that the token site that acknowledges the message receives the message from the source.

$$P_0 = \frac{(1-P_B)+\left[P_B\left(1-P_{L|B}\right)^{N+1}\right]}{(1-P_B)+\left[P_B\left(1-P_{L|B}\right)\right]} \tag{31}$$

For the case that $1 \le i \le N$, i neighbors miss the message from the source while N-i neighbors receive the message, given that the token site that acknowledges the message receives the message.

$$P_i = \frac{\binom{N}{i} P_B\left[\left(1-P_{L|B}\right)^{N-i+1}\right]\left(P_{L|B}\right)^i}{(1-P_B)+\left[P_B\left(1-P_{L|B}\right)\right]} \qquad , 1 \le i \le N \tag{32}$$

Figure 22 shows the state transition diagram for calculating S_i for $1 \le i \le N$. State x means x neighbors have not received the application-level message. S_i is the average number of steps needed to move from state i to state 0 or the expected number of steps before absorption if starting from state i. This expected number of steps can be calculated from transition probabilities $P_{n,m}$.

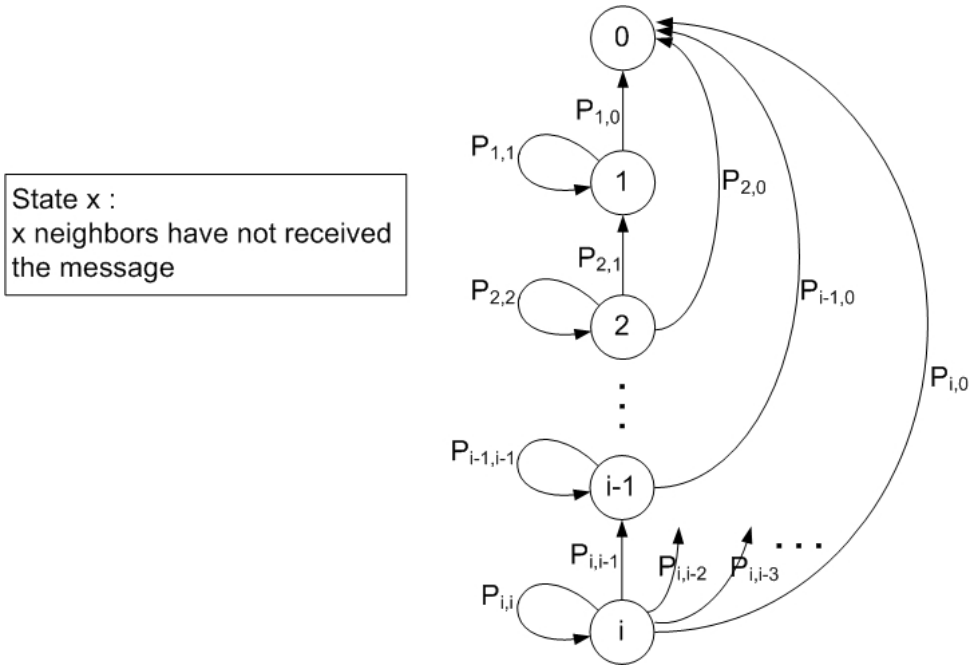

Figure 22. State transition diagram for calculating the average number of application-level messages retransmitted from the token site until all i neighbors that have missed the message receive it (S_i)

We will now show the calculation of $P_{n,m}$. We can move from state n to m when (n-m) neighbors successfully receive the retransmitted message and m neighbors lose the message. Therefore, $P_{n,m}$ is calculated as follows:

For m = 0,

$$P_{n.0} = (1 - P_B) + \left[P_B (1 - P_{L|B})^n \right] \tag{33}$$

For m > 0 and m ≤ n,

$$P_{n,m} = \binom{n}{m} P_B (1 - P_{L|B})^{n-m} (P_{L|B})^m \tag{34}$$

For m > n, $P_{n,m} = 0$.

As an example, if the source has 30 neighbors (N = 30) and vehicles independently lose a message ($\rho = 0$) with a probability $P_L = 0.01$, the average number of application-level messages retransmitted from the token site to the neighbors that miss the message, from the calculation, is 0.2633, which is close to the value 0.2572 from the simulation.

6.4.2.4 Average number of NACKs from the neighbors that miss the application-level message to request a message retransmission

As mentioned previously in section 6.4.2.3, after the source receives an ACK from the group and stops transmitting the application-level message, the number of neighbors that still have not received the message can be any value between 0 and N. Therefore, we calculate the average number NACKs transmitted from the neighbors that miss the message as follows:

Average number of NACKs transmitted from the neighbors that miss the message from the source

$$= \sum_{i=0}^{N} P_i(iR_i) \tag{35}$$

where

P_i is the probability that there are i neighbors that miss the message transmitted from the source.

R_i is the average number of NACKs that a neighbor that misses the message transmits until it receives the retransmitted message from the token site when there are total i neighbors that miss the message from the source. $R_i = 0$ when $i = 0$, and $R_i > 0$ otherwise.

Note that the average number of NACKs is calculated this way to make things simpler. The result from the calculation might be slightly higher than the actual average number of NACKs from the neighbors.

The calculation of P_i is the same as the one described in section 6.4.2.3. We will now describe the calculation of R_i for $1 \leq i \leq N$. Figure 23 shows the state transition diagram for calculating the average number of NACKs (R_i) that a neighbor A that misses the message transmits given that there are a total of i neighbors that miss the message from the source. The "done" state means that neighbor A successfully receives the retransmitted message from the token site; however, other neighbors may or may not have received the retransmitted message. The other state, x, where x = 1,2,…,i, means that x neighbors including neighbor A have not received the retransmitted message.

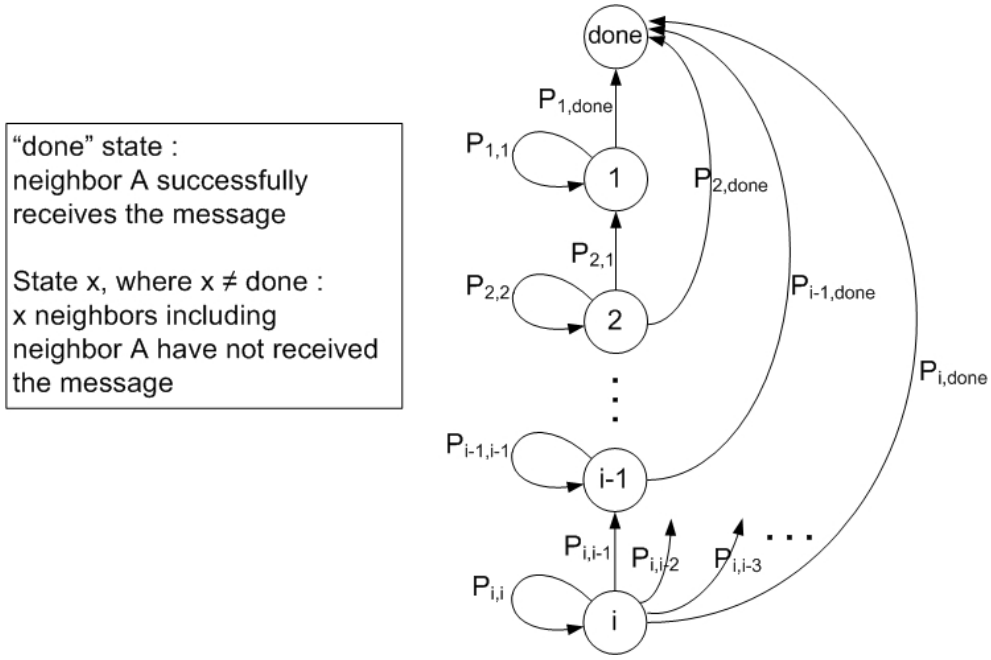

Figure 23. State transition diagram for calculating the average number of NACKs that neighbor A transmits until it receives the retransmitted message from the token site given that i neighbors miss the message from the source

R_i is the average number of steps needed to move from state i to the "done" state or the expected number of steps before absorption if starting from state i, which can be calculated from transition probabilities $P_{n,m}$. We can move from state n to state m that is < n, only if the token site hears at least one NACK from the n NACKs transmitted by n neighbors that have missed the message and therefore retransmits the message. Hence $P_{n,m}$ is calculated as follows:

For m = done,

$$P_{n,done} = \left[1 - \left(P_B P_{L|B}\right)^n\right] * \left\{\left(1 - P_B\right) + \left[P_B\left(1 - P_{L|B}\right)\right]\right\} \tag{36}$$

For m ≠ done and m < n,

$$P_{n,m} = \left[1 - \left(P_B P_{L|B}\right)^n\right]\binom{n-1}{n-m} P_B\left(1 - P_{L|B}\right)^{n-m}\left(P_{L|B}\right)^m \tag{37}$$

For m ≠ done and m = n,

$$P_{n,n} = \left(P_B P_{L|B}\right)^n + \left\{\left[1 - \left(P_B P_{L|B}\right)^n\right] P_B\left(P_{L|B}\right)^n\right\} \tag{38}$$

For m ≠ done and m > n, $P_{n,m} = 0$.

As an example, if the source has 30 neighbors (N = 30) and vehicles independently lose a message ($\rho = 0$) with a probability $P_L = 0.01$, the average number of NACKs transmitted from the neighbors that miss the application-level message, from the calculation, is 0.3053, which is close to the value 0.2989 from the simulation.

6.4.2.5 Average number of tokens transmitted from the token site until all group members receive it

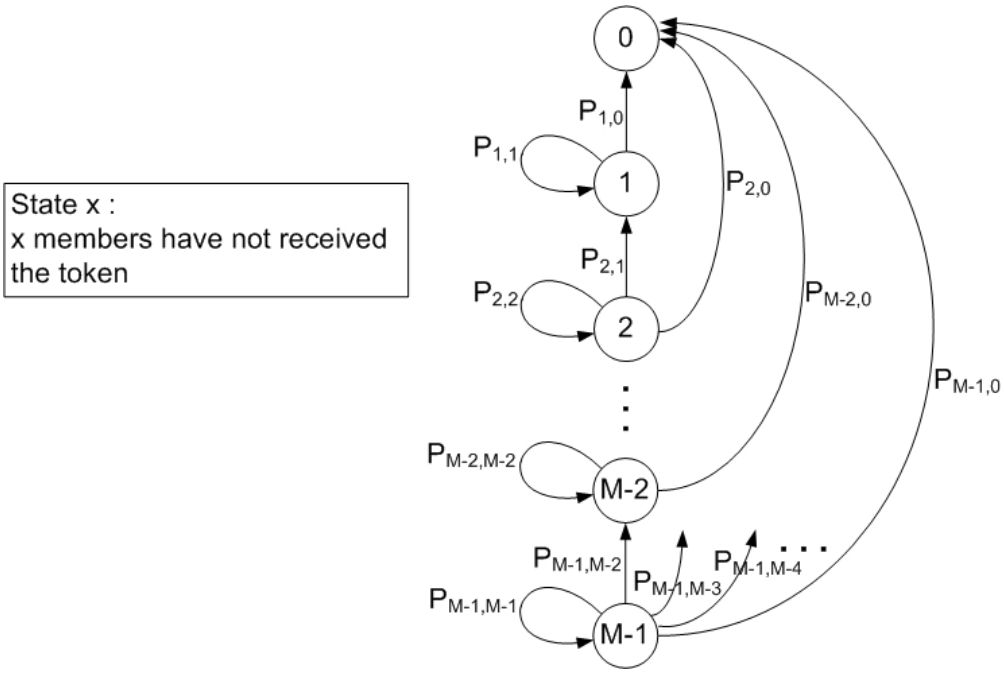

Figure 24. State transition diagram for calculating the average number of tokens transmitted from the token site until all M-1 members receive it

Figure 24 shows the state transition diagram for calculating the average number of tokens transmitted from the token site until all group members receive it. M is the total number of group members, so we need all M-1 members other than the token site to receive the token. State x means that x members have not received the token.

The average number of tokens transmitted from the token site is the average number of steps needed to move from state M-1 to state 0 or the expected number of steps before absorption if starting from state M-1. This expected number of steps can be calculated from transition probabilities $P_{n,m}$. The calculation of $P_{n,m}$ is the same as the one described in section 6.4.2.3.

As an example, if the group has 70 members (M = 70) and vehicles independently lose a message (ρ = 0) with a probability P_L = 0.01, the average number of tokens transmitted from the token site until all members receive it, from the calculation, is 1.5071, which is close to the value 1.5061 from the simulation.

6.4.2.6 Average number of NACKs from the group members that miss the token to request a token retransmission

The number of group members that miss the scheduled token can be any value between 0 and M-1, where M is the total number of group members. Therefore, we calculate the average number NACKs transmitted from the members that miss the token as follows:

Average number of NACKs transmitted from the members that miss the scheduled token

$$= \sum_{i=0}^{M-1} P_i(iR_i) \tag{39}$$

where

P_i is the probability that there are i members that miss the scheduled token.

R_i is the average number of NACKs that a member that misses the scheduled token transmits until it receives the retransmitted token from the token site when there are total i members that miss the scheduled token. $R_i = 0$ when $i = 0$, and $R_i > 0$ otherwise.

Note that the average number of NACKs from the members that miss the scheduled token is calculated this way to make things simpler. The result from the

calculation might be slightly higher than the actual average number of NACKs from the group members.

The calculation for P_i is as follows:

The case that i = 0 occurs when all M-1 members receive the scheduled token. So,

$$P_0 = (1 - P_B) + \left[P_B (1 - P_{L|B})^{M-1} \right] \tag{40}$$

For the case that $1 \le i \le M-1$, i from M-1 members do not receive the token while M-1-i members receive it. So,

$$P_i = \binom{M-1}{i} P_B \left[(1 - P_{L|B})^{M-1-i} \right] (P_{L|B})^i \quad , 1 \le i \le M-1 \tag{41}$$

We now describe how to calculate R_i for $1 \le i \le M-1$. Figure 25 shows the state transition diagram for calculating the average number of NACKs (R_i) that a member A that misses the scheduled token transmits given that there are total i members that miss the scheduled token. The "done" state means member A successfully receives the retransmitted token from the token site; however, other members may or may not have received the retransmitted token. The other state, x, where x = 1,2,…,i, means x members including member A have not received the token.

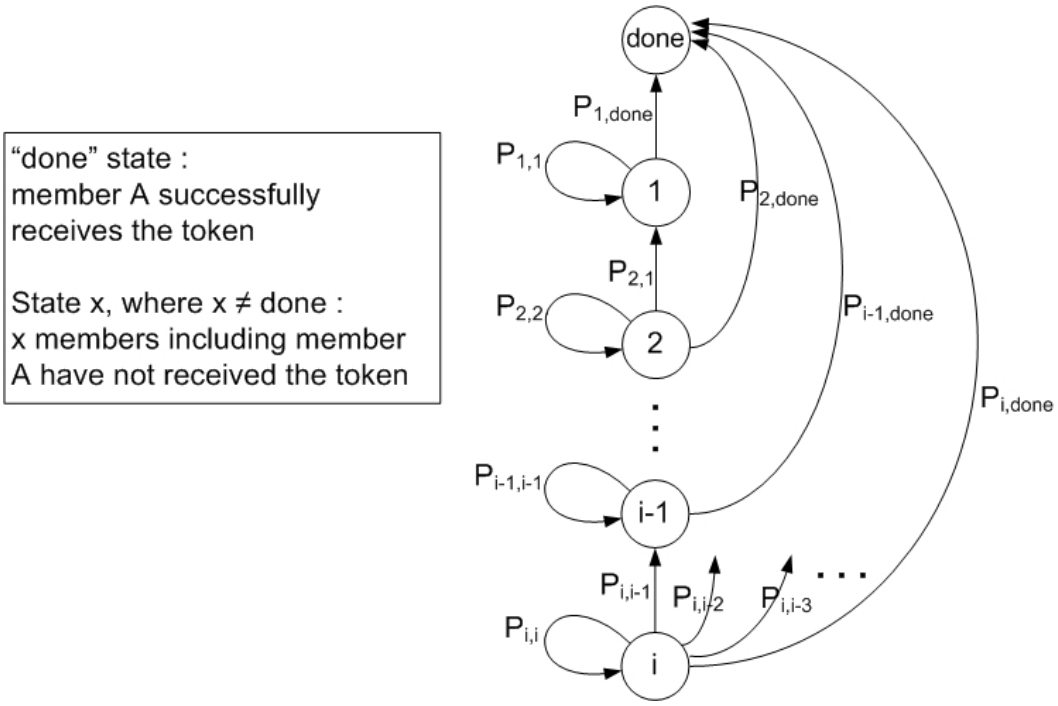

Figure 25. State transition diagram for calculating the average number of NACKs that a member A transmits (R_i) until receiving the retransmitted token from the token site given that a total of i members miss the scheduled token

R_i is the average number of steps needed to move from state i to "done" state or the expected number of steps before absorption if starting from state i, which can be calculated from transition probabilities $P_{n,m}$. The calculation of $P_{n,m}$ is the same as the one described in section 6.4.2.4.

As an example, if the group has 70 members (M = 70) and vehicles independently lose a message (ρ = 0) with a probability P_L = 0.01, the average number of NACKs transmitted from the group members that miss the scheduled token from the calculation is 0.7006, which is close to the value 0.6998 from the simulation.

6.4.3 Comparison of the average number of messages occurring from one application-level message transmission between using RNP and using the simple ARQ protocol

In this section, we compare the average number of messages that occur when an application-level message is transmitted between using RNP and using the simple ARQ protocol as described in section 6.4.1. In RNP, the average number of messages from one application-level message transmission is the sum of the average number of application-level messages transmitted from the source until it receives an ACK for the message as described in section 6.4.2.2, the average number of application-level messages retransmitted from the token site to the neighbors that miss the message as described in section 6.4.2.3, and the average number of NACKs transmitted from the neighbors that miss the application-level message to request for message retransmission as described in section 6.4.2.4. In the simple ARQ protocol, the average number of messages from one application-level message transmission is the sum of the average number of application-level messages transmitted from the source until it receives an ACK from each of the neighbors as described in section 6.4.1.2 and the average number of ACKs from all neighbors to the source as described in section 6.4.1.3.

We vary the number of neighbors of the source (N) from 0 to 30 and perform the comparisons for the three following cases:

1) The ideal case when there is no message loss ($P_L = 0$).

2) Show the effects of correlation of message loss (ρ). In this case, we set the probability that a vehicle loses a message (P_L) to 0.01 and consider the case when

vehicles independently lose a message ($\rho = 0$) and the case when the message loss at all receiving vehicles is fully correlated ($\rho=1$).

3) Show the effects of varying the probability that a vehicle loses a message (P_L). In this case, we set $\rho = 0$ (vehicles independently lose a message) and consider the cases where $P_L = 0.01$ and $P_L = 0.001$.

Case1: No message loss ($P_L = 0$)

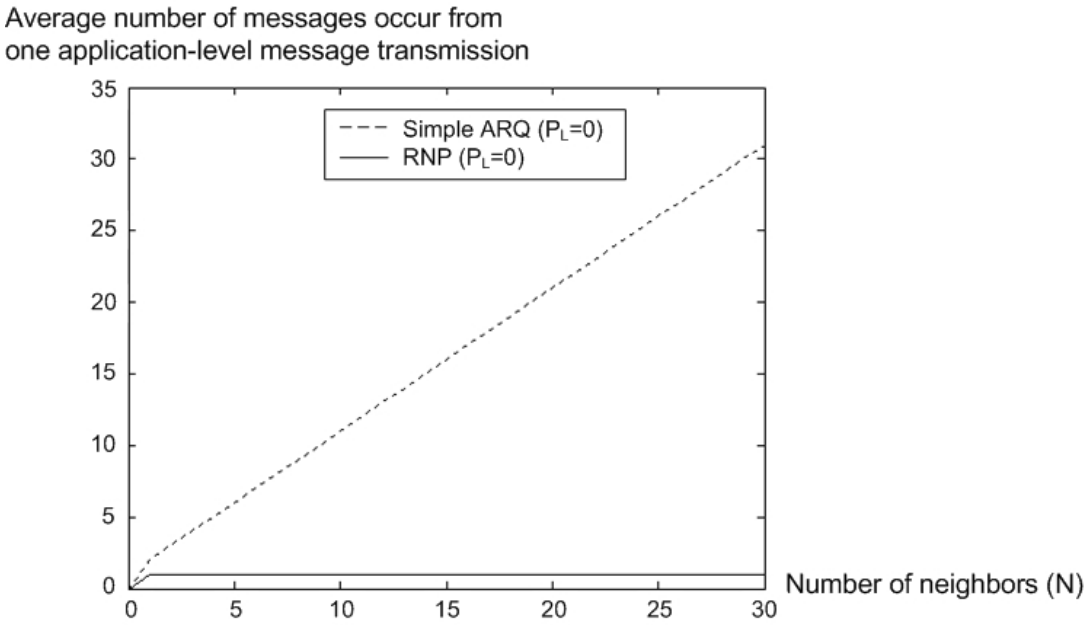

Figure 26. Comparison of the average number of messages occurring from one application-level message transmission between using RNP and using the simple ARQ protocol in case 1 (no message loss)

In the ideal case, where there is no message loss, there is only 1 message occurring from one application-level message transmission, independent of the number of neighbors (N), when RNP is used. There are N+1 messages transmitted when the simple ARQ protocol is used, as shown in Figure 26. The only message in RNP is the original application-level message transmitted from the source. Since there is no message loss, there are no NACKs to request message retransmission or retransmitted messages from

the token site. Note that in RNP, an ACK for the message is included in the scheduled M-RBP token, so it is not considered to be a message occurring from an application-level message transmission. In the simple ARQ protocol, each neighbor has to transmit an ACK for the message. Therefore, the number of messages occurring from one application-level message transmission increases linearly with the number of neighbors of the source (N).

Case2: The effects of correlation of message loss (ρ) when $P_L = 0.01$

Figure 27 shows the effects of correlation of message loss on the average number of messages that occur from one application-level message transmission between using RNP and using the simple ARQ protocol. Figure 28 shows the zoomed in results for RNP. When the message loss is fully correlated, the average number of messages occurring from one application-level message transmission is 1.0203, independent of the number of neighbors (N) when RNP is used, while the average number of messages when the simple ARQ protocol is used is much larger than RNP and also increases with the number of neighbors, as shown in Figure 27. The value of 1.0203 messages from RNP is the average number of application-level messages that the source has to transmit until it receives an ACK from the token site of the group. Since the message loss at all receiving vehicles is fully correlated, all members of the group, including all N neighbors of the source, also receive the application-level message from the source when the token site receives the message. Therefore, there are no NACKs to request for message retransmission or retransmitted message from the token site, regardless of the number of neighbors (N).

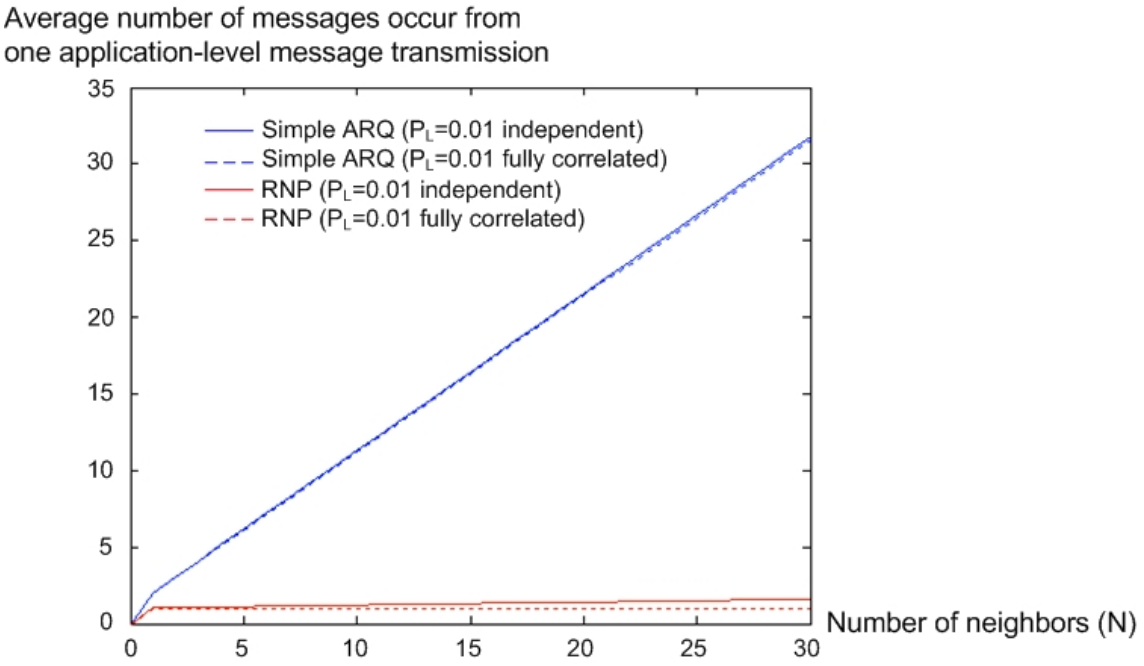

Figure 27. Comparison of the average number of messages occurring from one application-level message transmission between using RNP and using the simple ARQ protocol in case 2 ($P_L = 0.01$, $\rho = 0$ and 1)

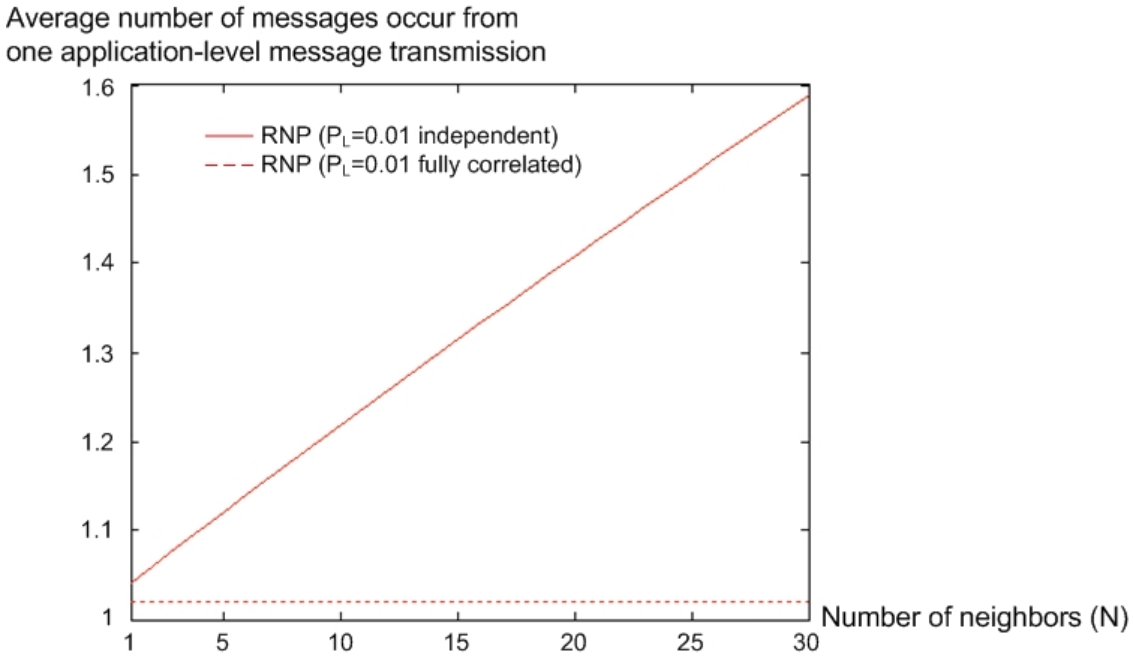

Figure 28. Average number of messages occurring from one application-level message transmission when using RNP in case 2 ($P_L = 0.01$, $\rho = 0$ and 1)

When vehicles independently lose a message (ρ = 0), the resulting average numbers of messages from both protocols are only slightly larger than the case in which the message loss is fully correlated (ρ = 1). For example, in the case of N = 30 and the simple ARQ protocol is used, the average number of messages is 31.7679 when ρ = 0, while it is 31.5791 when ρ = 1. For RNP, the average number of messages is 1.5889 when ρ = 0, while it is 1.0203 when ρ = 1.

In RNP, when vehicles independently lose a message, there are NACKs from the neighbors that miss the message from the source as well as retransmitted messages from the token site to these neighbors. The number of these NACKs and retransmitted messages increases with the number of neighbors, resulting in an average number of total messages from one application-level message transmission that increases with the number of neighbors as shown by the solid red line in Figure 28. However, the average number of messages from RNP increases at a much slower rate than that from the simple ARQ protocol.

Case3: The effects of varying the probability that a vehicle loses a message (ρ = 0, P_L = 0.01 and 0.001)

Figure 29 shows the effects of varying the probability that a vehicle loses a message (ρ = 0, P_L = 0.01 and 0.001) on the average number of messages occurring from one application-level message transmission when using RNP and when using the simple ARQ protocol. Figure 30 shows the zoomed in results for RNP. When P_L is 0.001, the average number of messages from both protocols is slightly smaller than when P_L is 0.01, as shown in Figure 29. For example, in the case of N = 30 when the simple ARQ protocol

is used, the average number of messages is 31.0884 when P_L is 0.001 and 31.7679 when P_L is 0.01. In the case of N = 30 when RNP is used, the average number of messages is 1.0617 when P_L is 0.001, while it is 1.5889 when P_L is 0.01. When P_L is 0.001, the average numbers of messages from both protocols increase with the number of neighbors. However, they increase at slower rates than when P_L is 0.01.

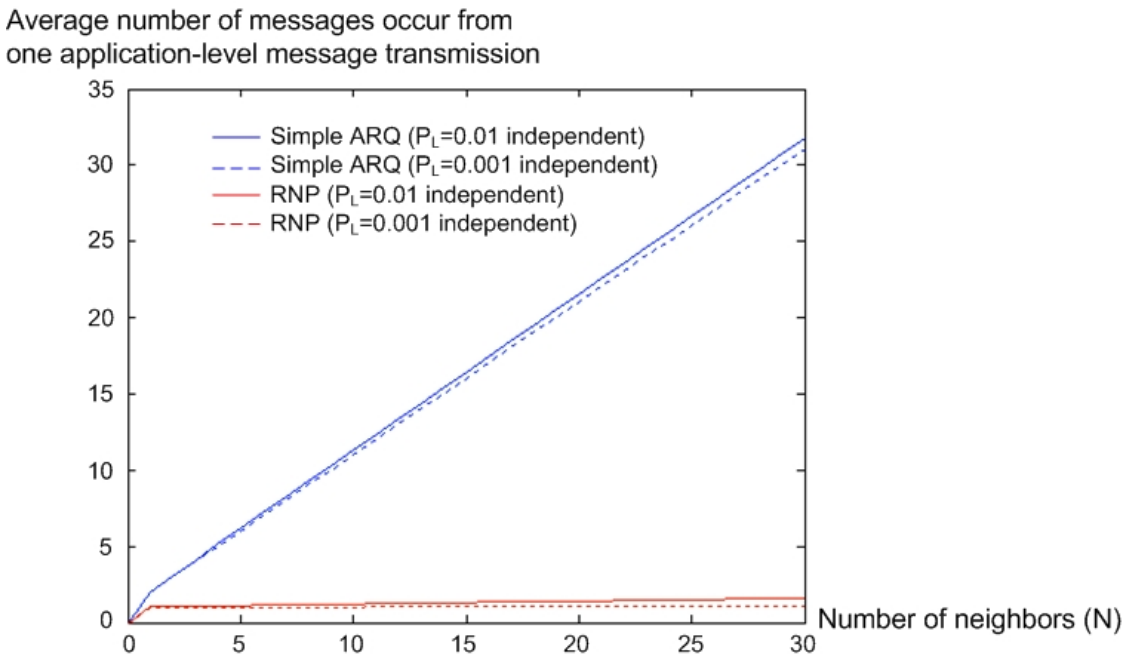

Figure 29. Comparison of the average number of messages occurring from one application-level message transmission between using RNP and using the simple ARQ protocol in case 3 (ρ = 0, P_L = 0.01 and 0.001)

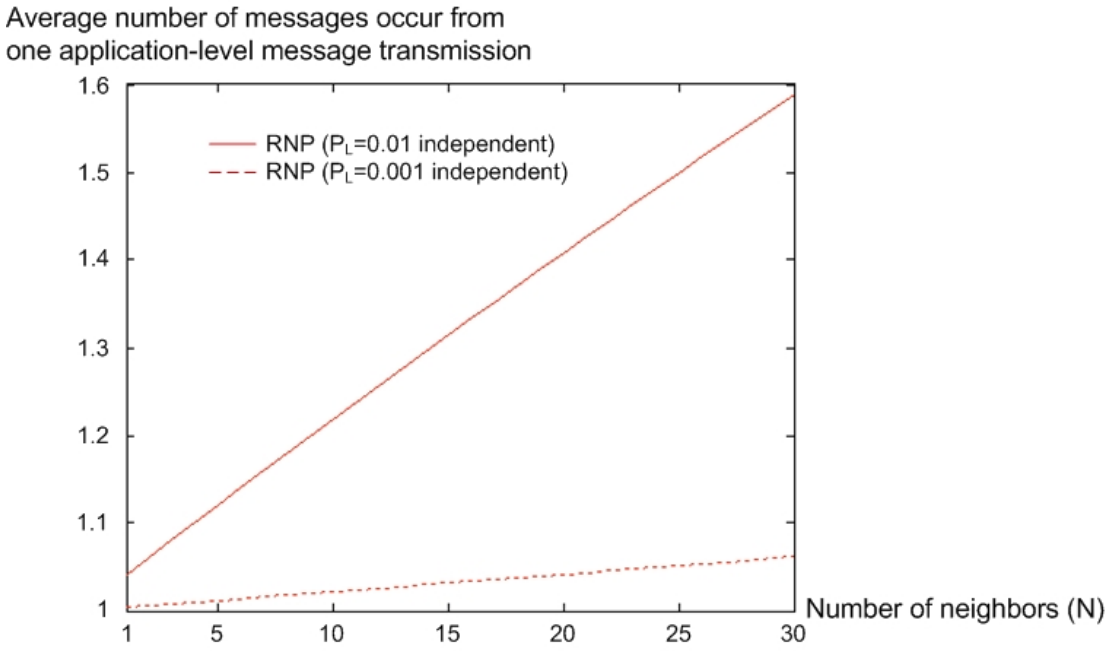

Figure 30. Average number of messages occurring from one application-level message transmission when using RNP in case 3 ($\rho = 0$, $P_L = 0.01$ and 0.001)

In summary, the average number of messages that occur from one application-level message transmission when using RNP is smaller than when using the simple ARQ protocol in all three cases. For the simple ARQ protocol, the average number of messages increases with the number of neighbors in all three cases. For RNP, the average number of messages is independent of the number of neighbors when there is no message loss or when the message loss at all vehicles is fully correlated, while the average number of messages increases with the number of neighbors when vehicles independently lose a message. However, in the case of independent message loss, the average number of messages from RNP still increases at a much slower rate than the average number of messages from the simple ARQ protocol.

Chapter 7

Application

This chapter shows an example of applying RNP in conjunction with sensors to avoid rear-end collisions. The goals of this chapter are to determine whether using RNP with sensors can increase highway capacity while improving safety, quantify the capacity improvement from using RNP with sensors, and compare the highway capacity improvement from using RNP with sensors with the case of using sensors alone. We show that both using RNP with sensors and using sensors alone can increase highway capacity; however, using RNP with sensors provides a much higher percentage increase in capacity. The increase in capacity is a function of the fraction of the vehicles that use a technology. If all of the vehicles are equipped with both sensors and RNP, the increase in highway capacity is 273.67% or the capacity is about 3.74 times the capacity when all vehicles have neither of the technologies. While in the case that all of the vehicles are equipped with sensors alone, the increase in highway capacity is 43.98% or the capacity is about 1.44 times the capacity when all vehicles have neither of the technologies.

To achieve these goals, we first propose two sets of rules that allow vehicles to automatically maintain safe following distances by using sensors alone and using sensors in conjunction with RNP. The percentage of vehicles equipped with sensors only, vehicles equipped with both sensors and communication devices, and vehicles that have neither of the technologies are varied, and the average safe inter-vehicle distance for each case is calculated. In turn, this distance is used to estimate the highway capacity.

The outline of this chapter is as follows. Section 7.1 provides an introduction and presents some related work on the impacts of collision avoidance technologies on highway capacity. Section 7.2 describes our proposed rules for using sensors alone and using sensors in conjunction with RNP to prevent rear-end collisions. Section 7.3 describes how to calculate the average safe inter-vehicle distance based on the proposed rules, and section 7.4 shows how to calculate highway capacity. The results are presented in section 7.5.

7.1 Introduction

Several automobile manufacturers are currently offering assisted driving systems to avoid rear-end collisions. One example is Adaptive cruise control (ACC) where sensors are used to prevent rear-end collisions [47]. A vehicle equipped with ACC uses onboard sensors to automatically adjust its speed in order to maintain a specified safe distance with the preceding vehicle. Vehicle-to-vehicle (v2v) communication can improve the safety of these systems by allowing vehicles to exchange information, such as speed and braking capability, in order to coordinate their operation.

While safety is the primary consideration in many automated systems, we must also consider how they will affect the capacity of a highway. Since not all vehicles on a highway will be equipped with new technologies, the impacts should be assessed for different portions of equipped vehicles.

The impact of ACC on highway capacity has been studied. In [48] a mix of vehicles that are automatically controlled by an ACC system and manually controlled on

the on-ramp of a highway is studied. The rules for ACC vehicles to automatically maintain the safe following distance and merge onto the highway are described. The results show a 33% increase in capacity of vehicles that can safely enter the highway when all vehicles are equipped with ACC.

In [21], a Cooperating ACC (CACC) system, which allows vehicles to communicate with each other, is investigated. They vary the proportion of manually operated vehicles, ACC vehicles, and CACC vehicles. The results show that ACC vehicles provide, at most, a 7% increase in highway capacity while CACC vehicles can double the capacity of the highway. This work is similar to what we do in this chapter, but their system requires driver intervention for rapid deceleration, while our system assumes automatic braking.

In [22], the highway capacity increase from IntelliDrive (which also allows vehicles to communicate with each other) is studied. They show that the increase in highway capacity with their control rules is between 20% and 50%.

RNP allows each vehicle to reliably communicate with all of its neighbors within a specified distance in the same lane of it. This is different from the related work; which only allows vehicles to communicate with their immediate neighbors. Communicating with all of the vehicles in a neighborhood provides a faster response to a situation; which is similar to drivers who respond to situations that are several vehicles away rather than just observing the vehicle in front of them.

7.2 Rules for using sensors and RNP to avoid rear-end collisions

In section 7.2.1, the different types of vehicles on a highway are described. Then the rules for using sensors alone and using sensors in conjunction with RNP to avoid rear-end collision are presented in sections 7.2.2 and 7.2.3 respectively.

7.2.1 Types of vehicles on a highway

We categorize vehicles on highway into three types as follows:

1) Manual Vehicles; which have neither sensors nor communication and are manually controlled.

2) Vehicles with Sensors; which have onboard sensors but no communication devices and are automatically controlled according to the rules in section 7.2.2.

3) Communicating Vehicles; which have both sensors and communication devices. This type of vehicle uses RNP as the communication protocol and is automatically controlled according to the rules in section 7.2.3.

7.2.2 Rules for vehicles with sensors

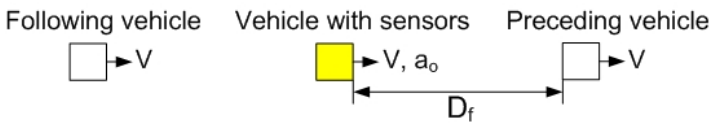

Figure 31. Variables related to the rules for vehicles with sensors

Figure 31 shows variables related to the rules for vehicles with sensors. All vehicles on a highway are assumed to move at the same speed, V km/h. Each vehicle has its own maximum deceleration rate (a_o) in m/s^2. D_f is the safe following distance in meters that the vehicle with sensors maintains to the preceding vehicle.

D_f is calculated by assuming that the maximum deceleration rates of vehicles are between a maximum deceleration rate a_{max} and a minimum deceleration rate a_{min}. In order to avoid collisions with the preceding vehicle, we assume the worst-case scenario; in which the preceding vehicle can decelerate at the maximum deceleration rate a_{max}. We assume that we monitor and know our own deceleration rate a_o, under the current road and load conditions. In our results, we use a_{min} = -5 m/s^2 and a_{max} = -8.5 m/sec^2, as in [49].

The vehicle with sensors always maintain D_f, which is calculated from its perception-reaction time and the difference in the deceleration rate between itself and the preceding vehicle:

$$D_f = (T_s*V/3.6) + [V^2/(25.92|a_o|)] - [V^2/(25.92|a_{max}|)] \quad (42)$$

where T_s is the delay until a vehicle with sensors detects an emergency situation (when its preceding vehicle suddenly brakes) and the brake is automatically applied. This delay includes the mechanical response time of an automobile braking system.

7.2.3 Rules for communicating vehicles

Each communicating vehicle uses RNP to communicate with its neighboring vehicles. The upper bound of the delay until RNP guarantees that a message from a

vehicle is successfully received and committed by all of its neighbors is D_{max}, which is calculated in chapter 6. The token passing mechanism used in RNP allows vehicles to know which of their neighbors can communicate.

Communicating vehicles use communications to exchange information on their deceleration rates and to provide a notification of an emergency stop. By exchanging information on deceleration rates, communicating vehicles do not have to assume the worst-case deceleration rates when calculating the safe following distance D_f that they have to maintain to their preceding vehicles. Communicating vehicles notify their neighbors when an emergency stop occurs by using RNP to transmit a warning message. By doing this, the time to detect a physical change in the operation of the preceding vehicle is reduced and a vehicle can respond to a stop by a vehicle several vehicles in front, as human drivers.

A communicating vehicle negotiates and uses a group deceleration rate, a_c, for decelerating instead of its actual maximum deceleration rate, a_o. The negotiated rate, a_c, is the minimum deceleration rate among the group of neighboring communicating vehicles without any vehicles with sensors or manual vehicles between them. If there is an intervening manual or sensor controlled vehicle, the communicating vehicles near it will assume the minimum deceleration rate.

Figure 32 shows an example of how communicating vehicles choose the deceleration rates a_c to be used. Vehicles labeled with C are communicating vehicles; while vehicles labeled with M are manual vehicles or vehicles with sensors.

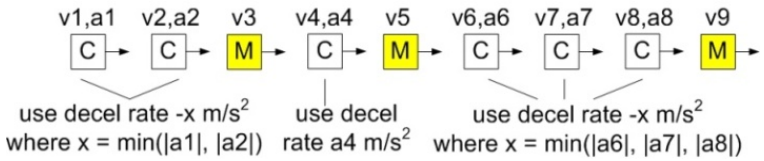

Figure 32. Negotiated deceleration rates that communicating vehicles choose to use

Similar to the sensor case, each communicating vehicle maintains the safe following distance D_f to its preceding vehicle. However, the calculation of D_f depends on whether or not the nearby vehicles can communicate as follows.

Case1: Neither the preceding nor following vehicle can communicate

In this case, the communicating vehicle has to rely on the information from its sensors. The vehicle uses its actual maximum deceleration rate a_o in this case because both the vehicle in front of it and the vehicle behind it are not communicating vehicles. Therefore, the calculation for D_f in this case is exactly the same as the one in section 7.3.2. The equation is repeated here for convenient.

$$D_f = (T_s*V/3.6) + [V^2/(25.92|a_o|)] - [V^2/(25.92|a_{max}|)] \quad (43)$$

Case 2: The preceding vehicle cannot communicate, but the following vehicle can

In this case, the communicating vehicle also has to rely on its sensors. However, it will use the negotiated deceleration rate a_c instead of the actual rate a_o because the following vehicle is also a communicating vehicle. So,

$$D_f = (T_s*V/3.6) + [V^2/(25.92|a_c|)] - [V^2/(25.92|a_{max}|)] \quad (44)$$

Case 3: The preceding vehicle can communicate

In this case, the communicating vehicle and its preceding vehicle agree to use the same negotiated rate a_c and the communicating vehicle can detect an emergency situation within D_{max} second after the preceding vehicle uses RNP to transmit a warning message. The perception-reaction time of a communicating vehicle (T_c) is the sum of the delay until it detects the situation (D_{max}) and the delay until the brake is automatically applied. Therefore,

$$D_f = T_c * V/3.6 \quad (45)$$

7.3 Average safe inter-vehicle distance calculation

This section shows how to calculate the average safe inter-vehicle distance that ensures no collisions with preceding vehicles. Each vehicle needs to maintain different safe following distances depending on the types of itself, its preceding vehicle, and its following vehicle. Therefore, the percentage of each type of vehicles on a highway affects the average safe inter-vehicle distance. In this section, we first present the equation for calculating the average safe inter-vehicle distance ($\overline{D}$), followed by the details on how to calculate the average safe following distance that each vehicle has to maintain in different cases.

The average safe inter-vehicle distance $\overline{D}$ is calculated as:

$$\overline{D} = (P_m * D_m) + (P_s * D_s) + (P_c * D_c) \quad (46)$$

P_m, P_s, and P_c are the probability that a vehicle is a manual vehicle, a vehicle with sensors, and a communicating vehicle respectively, where $P_m + P_s + P_c = 1$. D_m, D_s, and D_c are the average safe following distance that manual vehicles, vehicles with sensors, and communicating vehicles maintain to their preceding vehicles respectively.

The average safe following distances D_m, D_s, and D_c are calculated as follows.

D_m calculation

According to [21, 50], the time gaps that manual drivers maintain with their preceding vehicles are assumed to be normally distributed with a mean of 1.1 second and standard deviation of 0.15 second. This value was taken from a statistical analysis of the UMTRI ACC FOT baseline case human driving data. We will assume that a time gap of 1.1 second is safe and adequate for manual drivers to stop their vehicles without colliding with the preceding vehicles. So, the average safe following distance for manual vehicles is:

$$D_m = 1.1*V/3.6 \quad (47)$$

D_s calculation

A vehicle with sensors uses Eq.(42) in the rules in section 7.2.2 to calculate the safe following distance that it has to maintain. Assuming that the maximum deceleration rate a_o of a vehicle is uniformly distributed over the interval $[a_{max}, a_{min}]$, the average safe following distance that vehicles with sensors maintain is:

$$D_s = (T_s*V/3.6) + \{V^2 * ln(|a_{max}|/|a_{min}|) / [25.92* (|a_{max}|-|a_{min}|)]\} - [V^2/(25.92|a_{max}|)] \quad (48)$$

D_c calculation

A communicating vehicle maintains different safe following distances in 3 different cases according to the rules described in section 7.2.3. Therefore, the average safe following distance that communicating vehicles maintain is

$$D_c = (P_1 * D_{c1}) + (P_2 * D_{c2}) + (P_3 * D_{c3}) \quad (49)$$

where P_1, P_2, and P_3 are the probability that case 1, 2, and 3 occur respectively. D_{c1}, D_{c2}, and D_{c3} are the average safe following distance that communicating vehicles maintain in case 1, 2, and 3 respectively. The probability and the average safe following distance for each case are calculated as follows.

Case1: Neither the preceding nor following vehicle can communicate

The probability that case 1 occurs is $P_1 = (P_m+P_s)^2$. In this case, the communicating vehicle maintains the same safe following distance as a vehicle with sensors does. Therefore,

$$D_{c1} = D_s = (T_s * V/3.6) + \{V^2 * ln(|a_{max}|/|a_{min}|) / [25.92* (|a_{max}| - |a_{min}|)]\} - [V^2/(25.92|a_{max}|)] \quad (50)$$

Case 2: The preceding vehicle cannot communicate, but the following vehicle can

The probability that case 2 occurs is $P_2 = (P_m+P_s)*P_c$. In this case, the communicating vehicle uses Eq.(44) to calculate the safe following distance. Therefore,

$$\begin{aligned} D_{c2} &= (T_s * V/3.6) + \int_{|a_{min}|}^{|a_{max}|} f(x) * \left(\frac{V^2}{25.92x}\right) dx - [V^2/(25.92|a_{max}|)] \\ &= (T_s * V/3.6) - [V^2/(25.92|a_{max}|)] + \\ &\left(\frac{nV^2}{25.92*(|a_{max}| - |a_{min}|)^n}\right) \int_{|a_{min}|}^{|a_{max}|} \left(\frac{(|a_{max}| - x)^{n-1}}{x}\right) dx \end{aligned} \quad (51)$$

where X is |the negotiated deceleration rate that the communicating vehicle we are considering chooses to use|, f(x) is the probability density function of X, and n is the average number of communicating vehicles (including the vehicle we are considering) that agree to use the same negotiated rate as the vehicle we are considering.

We now show the details on how f(x) and n are calculated. The negotiated deceleration rate depends on the number of communicating vehicles in a row behind the vehicle we are considering and the actual deceleration rates of these communicating vehicles. f(x) can be calculated from F(x), the cumulative distribution function of X, as follows.

F(x) = 1 – P (the vehicle we are considering and all n-1 communicating vehicles in a row behind it have |actual maximum deceleration rate| > x)

$$= 1 - \left(\frac{|a_{max}| - x}{|a_{max}| - |a_{min}|}\right)^n$$

$$f(x) = \left(\frac{n}{|a_{max}| - |a_{min}|}\right) * \left(\frac{|a_{max}| - x}{|a_{max}| - |a_{min}|}\right)^{n-1} \quad (52)$$

In our case, the vehicle we are considering and the following vehicle are communicating vehicles; therefore, n is calculated as follows.

$$n = \sum_{i=2}^{\infty} i(P_c{}^{i-2})(1 - P_c) = \sum_{i=0}^{\infty}(i+2)\left(P_c{}^{i}\right)(1 - P_c) \quad (53)$$

Case 3: The preceding vehicle can communicate

The probability that case 3 occurs is $P_3 = P_c$. In this case, the communicating vehicle uses Eq.(45) to calculate the safe following distance. Therefore,

$$D_{c3} = T_c * V/3.6 \quad (54)$$

To summarize, Eq.(49)-Eq.(54) can be used to calculate D_c, then the average safe inter-vehicle distance $\overline{D}$ can be calculated from D_m, D_s, and D_c as shown in Eq.(46).

7.4 Highway capacity calculation

Reference [51] defines the capacity of a facility as the maximum hourly rate at which persons or vehicles reasonably can be expected to traverse a point or a uniform section of a lane or roadway during a given time period under prevailing roadway, traffic, and control conditions. From this definition, highway capacity (C) in vehicles/hour/lane can be estimated as

$$C = \frac{3600}{\overline{T} + T_L} = \frac{3600}{\left(\frac{3.6 * \overline{D}}{V}\right) + \left(\frac{3.6 * l}{V}\right)} \tag{55}$$

where

$\overline{D}$ is the average safe inter-vehicle distance in meters calculated in section 7.3.

l is the average vehicle length in meters.

$\overline{T}$ is the average safe time gap in seconds (3.6* $\overline{D}$/V).

T_L is the time that a vehicle covers the distance equal to the average vehicle length l (T_L = 3.6*l/V).

7.5 Results

This section shows the resulting average safe inter-vehicle distance and highway capacity for three cases. In the first case, the percentage of each of the three types of vehicle is varied, but the speed of vehicles (V) is fixed at 100 km/h (62.14 mph). The second case is the same as the first case except that there are only two types of vehicle on a highway i.e. manual vehicles and either communicating vehicles or vehicles with sensors. In the third case, the vehicle speeds are varied from 0 to 120 km/h (74.56 mph), but there is only one type of vehicle on a highway.

The following assumptions are used in all three cases. All vehicles move with the same speed. a_{min} and a_{max} are -5 m/s^2 and -8.5 m/s^2 respectively [49]. Manual drivers maintain the average time gap of 1.1 second as in [21], T_s is 0.245 second as in [48], and T_c is 0.181 second. T_c is the sum of the maximum delay D_{max} from RNP and the delay until the brake is applied. D_{max} is 0.081 second as shown in the example in section 6.3.3. This value is the maximum delay until RNP guarantees that a message from a vehicle is successfully received and committed by all of its neighbors within 125 meters from it assuming that the average vehicle length (l) is 4.3 meters [45]. The brake delay is 0.1 second as in [46].

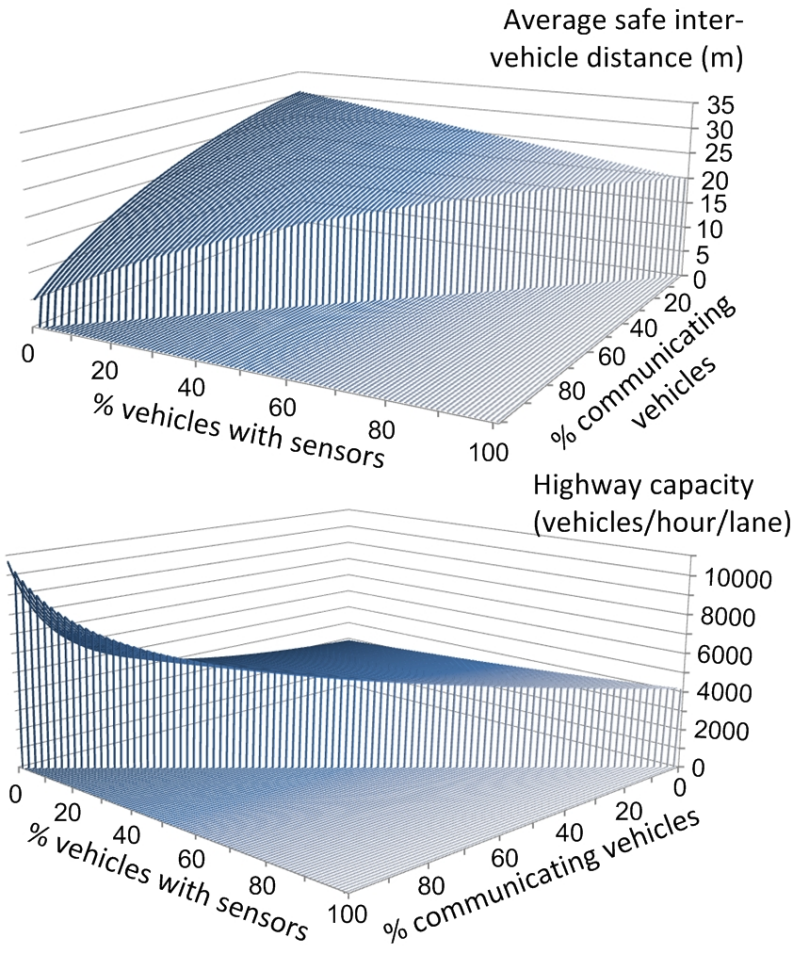

Figure 33. Average safe inter-vehicle distance and estimated highway capacity at speed 100 km/h when percentages of the three vehicle types are varied

Both vehicles with sensors and communicating vehicles help decrease the average safe inter-vehicle distance; however, communicating vehicles provide a much higher percentage of improvement than vehicles with sensors. Figure 33 shows the average safe inter-vehicle distance and estimated highway capacity for the first case. The average safe distance when 100% of the vehicles on a highway are manual vehicles is 30.5556 meters. The average distance is decreased when there is at least one vehicle with sensors or one communicating vehicle on a highway. When 100% of the vehicles are communicating

vehicles, the average distance is only 5.0278 meters, a 83.54 % decrease from the 100% manual case. On the other hand, when 100% of the vehicles are vehicles with sensors, the average distance is 19.9078 m, a 34.85% decrease. Note that these distances ensure no collisions with preceding vehicles if the time gap of 1.1 second is adequate for human drivers to stop their vehicles without colliding with the preceding vehicles.

The estimated highway capacity is increased when there are vehicles with sensors and/or communicating vehicles on a highway; however, communicating vehicles provide a much higher percentage of improvement than vehicles with sensors. Both vehicles with sensors and communicating vehicles help decrease the average safe inter-vehicle distance, thus help increase the capacity according to Eq.(55). In Figure 33, when 100% of the vehicles are manual vehicles, the capacity is 2868.98 vehicles/hour/lane. When 100% are vehicles with sensors, the capacity is increased to 4130.9 vehicles/hour/lane. This is a 43.98% increase from 100% manual case. On the other hand, when 100% are communicating vehicles, the capacity is increased tremendously to 10720.64 vehicles/hour/lane. This is about 3.74 times the capacity of a highway with manual vehicles or 273.67% increase from 100% manual case.

The 43.98% increase in capacity for the 100% sensor case is different from the results in [48] and [21] because of different assumptions and ways of using sensors. [48] assumes that the expected speed of vehicles with sensors and manual vehicles are 120 and 110 km/h respectively and focuses on the on-ramp traffic merging onto a highway. [21] requires the vehicles with sensors to maintain a fixed large time gap of 1.4 second from the preceding vehicles to allow their drivers to intervene in emergency situations, so their percentage increase in capacity is smaller than ours.

In the case of 100% communicating vehicles, our results show a much higher percentage increase in capacity than [21] and [22]. This is due to small message delivery delay provided by RNP; which results in T_c of only 0.181 second. In addition, since all vehicles choose to use the same deceleration rate in this case, we do not need the part of the safe distance due to the difference in the deceleration rates.

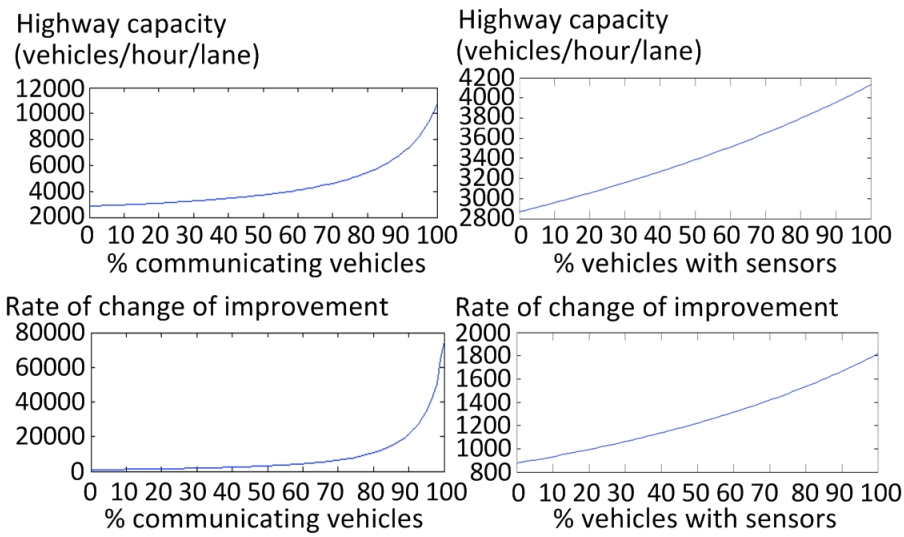

Figure 34. Rate of change of improvement in capacity at speed 100 km/h when the percentage of communicating vehicles/vehicles with sensors is varied

Figure 34 shows the highway capacity and the rate of change of improvement in capacity at speed 100 km/h for the second case. The left hand side is when there are only manual vehicles and communicating vehicles on a highway; while the right hand side is when there are only manual vehicles and vehicles with sensors. The rate of change of improvement is higher as the percentage of communicating vehicles/vehicles with sensors increases. In the case of communicating vehicles, the capacity improves very slowly when the percentage of communicating vehicles is about 30% or less, then it increases with a little higher rate until the percentage is about 85%, and improves very quickly after this point. In the case of vehicles with sensors, the rate of change of

improvement is almost linear; however, it is much lower than in the case of communicating vehicles.

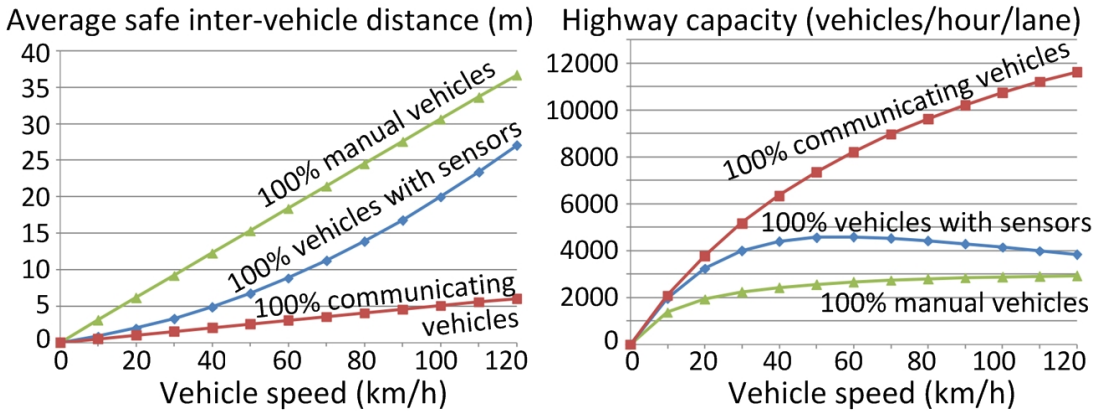

Figure 35. Average safe inter-vehicle distance and estimated highway capacity when vehicle speed is varied

Figure 35 shows the average safe inter-vehicle distance and estimated highway capacity for the third case. The capacity increases when the vehicle speed increases in the case of 100% manual vehicles and 100% communicating vehicles, while in the case of 100% sensor vehicles, the capacity increases until the point that the speed is 57.29 km/h and then decreases. This is because the average safe time gaps ($\overline{T}$) in both 100% manual vehicles and 100% communicating vehicles are constant at 1.1 second and 0.181 second respectively. Therefore, as shown in Eq.(55), the capacity increases with the vehicle speed because a vehicle takes shorter time T_L to cover the distance l when the vehicle speed increases. On the contrary, in the case of 100% sensor vehicles, the average safe time gap ($\overline{T}$) increases with the vehicle speed because of the part of the average safe inter-vehicle distance ($\overline{D}$) due to the difference in deceleration rates (the second and third parts of Eq.(48)). After speed 57.29 km/h, the increase in the average safe time gap

$\overline{T}$ is greater than the decrease in the time T_L that a vehicle takes to covers the distance *l*, and thus results in the decrease in highway capacity. The maximum capacity for 100% sensor case is 4538.48 vehicles/hour/lane.

To summarize, both using sensors alone and using RNP in conjunction with sensors can help increase both safety and highway capacity; however, using RNP with sensors can provide a much higher percentage increase in capacity than using sensors alone. Safety is improved because both methods allow vehicles to maintain safe following distances, while human drivers may follow preceding vehicles too closely. Highway capacity is improved because both methods provide faster reaction time than human drivers, which results in smaller safe following distances and an increase in highway capacity. In addition, when using RNP with sensors, neighboring vehicles can exchange information about their deceleration rates and choose to use the same safe deceleration rate. By doing this, vehicles do not need to maintain the part of the safe following distance due to the difference in deceleration rates as in the case of using sensors alone, which results in an even higher percentage increase in highway capacity.

The capacity improvement in both the case of using RNP with sensors and using sensors alone increases as the percentage of equipped vehicles on a highway increases. However, the rates of change of capacity improvement in the two cases are different. When sensors are used alone, the capacity improves almost linearly as the portion of equipped vehicles increases. In contrast, when RNP is used along with sensors, the highway capacity improves slowly when there is a small portion of equipped vehicles on a highway, and then improves more and more quickly as the portion of equipped vehicles increases. This is because we have to communicate with nearby vehicles to benefit from

communication protocols. Therefore, communication protocols such as RNP will provide substantially benefits when a large portion of vehicles are equipped with communication devices.

Chapter 8

Conclusion

The goals of this dissertation are to explore a new communication paradigm for VANETs, called neighborcast, and to develop a communication protocol, called reliable neighborcast protocol (RNP), to implement the paradigm. We are interested in VANET applications that improve road safety by avoiding collisions and have also shown an example of using RNP in conjunction with sensors to avoid rear-end collisions.

In chapter 2, we have introduced a new communication paradigm, neighborcast, and neighborhood concept, on which RNP is developed. In neighborcast, a vehicle communicates with all of its neighbors i.e., all vehicles within the neighborhood span 2L around it. Each vehicle's set of neighbors is usually different from that of nearby vehicles. Neighborcast can be considered as a special case of multicast where each vehicle communicates with a subset of nearby vehicles; however, the implementation of reliable multicast/broadcast protocols and our reliable neighborcast protocol, RNP, are significantly different. In a reliable multicast/broadcast protocol, all communicating vehicles are in one group. But in a reliable neighborcast protocol, the group size is constrained to limit the communication delay, so we cannot have all vehicles in one group. As a result, each vehicle may communicate in more than one overlapping group.

Neighborcast is useful for VANET applications that require communications with nearby vehicles in order to coordinate movement; RNP is created to provide reliable neighborcast communications for VANETs. The guarantees that RNP provides include

guaranteed message delivery from each vehicle in a VANET to all of its neighbors within a bounded delay, ensuring that all the neighbors that receive the same messages sequence them in the same order and use each of them at the same time, and providing the neighbors the knowledge of whether all the other neighbors have received the message or which neighbors are missing the message.

We have shown how to create RNP as an overlay protocol on top of the modified version of a recently invented reliable broadcast protocol, M-RBP. M-RBP provides guaranteed message delivery to all members in a broadcast group and guarantees that all the members that receive the same message commit the message at the same time. RNP is composed of two parts. The first is the self-organizing protocol that organizes vehicles into broadcast groups that use the modified version of M-RBP. The self-organizing protocol ensures that each vehicle is always a member of at least one broadcast group that contains itself and all of its neighbors. This way, it can reach all of its neighbors by transmitting messages in one broadcast group, resulting in the same message sequencing for all neighbors. The second part is the mechanism that provides the RNP guarantees. It transfers the guarantees provided by the modified version of M-RBP from the broadcast group level to the neighborhood level.

We have created a self-organizing protocol that is described in chapter 3. The self-organizing protocol is designed to create overlapping broadcast groups where each group moves with its members and adjacent groups overlap by at least a target overlap size. The minimum value of the target overlap size is $2L + 2\Delta + e$, where $2L$ is the neighborhood span, 2Δ is the extra overlap to give vehicles time to join a new group from either edge, and e is the extra overlap to give a vehicle in the overlap time to inform other

vehicles about the overlap and ensures that the overlap is always $\geq 2L + 2\Delta$. We have shown that this target overlap size ensures that each vehicle in the network can always reach all of its neighbors in one group and is always accepted to a new group before it needs to use the new group to communicate with its neighbors.

Our self-organizing protocol is a distributed protocol in which each member of a group independently executes the rules, proposes a change to the location of the group, and changes the location of the group. The protocol ensures that all members of the group always perform the same change to the group at the same time by allowing each member to include its proposed change in its scheduled M-RBP token or a new source message for the group, and the change does not take place until the token or the source message is voted in and committed according to the M-RBP voting mechanism. To reduce the frequency that vehicles need to join a new group, each group member is responsible for moving the edges of the group with the median speed of the group members, while maintaining the target overlap size with adjacent group behind it (if any). To limit the delay until a message is committed, each member is also responsible for splitting the group when the group span is large enough to fit two groups that overlap with each other by the target overlap size into one group. To limit the number of groups of which each vehicle is a member to two at most, each member is responsible for merging its group with the adjacent group when the group overlaps the center of the adjacent group. Each member is also responsible for starting new groups when necessary and leaving groups that no longer cover its position.

We have evaluated the performance of the self-organizing protocol in chapter 4. We have calculated the upper bound of the average number of times that a vehicle joins a new group per minute (J_{max}), verified the calculation result with the simulation, and compared the result with stationary groups. As expected, our self-organizing protocol results in a smaller J_{max}, which, in turn, results in fewer join request messages and recovery messages for the join requests compared to stationary groups. For example, when vehicles move with speeds between 60 and 80 mph, and each vehicle has the same neighborhood span (2L) of 250 m, J_{max} of our self-organizing protocol is 2.15 times/min, while J_{max} of the stationary groups approach is 8.58 times/min. This is because our protocol moves the edges of the groups with group members, rather than maintaining stationary groups which need to be left and rejoined when vehicles cross the group's boundaries.

We have also calculated the upper bound and lower bound of the average number of groups of which a vehicle is a member and verified the results with the simulation. For example, when 33 vehicles moving with constant speeds between 60 and 80 mph are initially put in 3 broadcast groups, each group has the same target group span of 625 m, adjacent groups overlap by the target overlap size of 252.74 m, and each vehicle has the same neighborhood span (2L) of 250 m, the average number of groups of which a vehicle is a member is 1.7, which is between the calculated lower bound of 1.67 and the upper bound of 2. This confirms that our self-organizing protocol can limit the average number of groups of which a vehicle is a member to two at most.

The simulations have been performed to study the effects of message loss on the self-organizing protocol. The results show that even in an environment with message loss, in which the probability that a vehicle loses a message is 0.01 and 0.001, the protocol always ensures that each vehicle is always a member of at least one broadcast group that contains itself and all of its neighbors at all times.

We have created the mechanism that provides the RNP guarantees, which is the second part of RNP and is presented in chapter 5. To ensure that all neighbors that receive the same messages sequence them in the same order and use each of them at the same time, each vehicle transmits application-level messages to all of its neighbors in only one broadcast group and vehicles order received application-level messages based on the times the messages are committed. To allow only the neighbors of a source to recover, vote, and commit an application-level message from the source, the source includes its current information in the message. This information is transferred to the token that acknowledges the message, which is then recovered by all group members. At the start of the message recovery period, all members use this information to decide whether or not to recover the message by estimating the current position of the source to determine the total number of the source's neighbors and whether they are among these neighbors or not. Only the neighbors of the source, rather than all group members, vote for the message, and the message is committed if more than half of the neighbors, rather than half of the group members, voted that they have received the message. A neighbor can know whether all of the other neighbors have received the message, or which neighbors are missing it, by looking at the votes it has received from other neighbors. All neighbors from which it has received YES votes have successfully received the message, while all

neighbors from which it has received NO votes missed the message. Neighbors it has not received votes from may have missed the message.

We have evaluated the performance of RNP in chapter 6. The upper bound of the delay until RNP guarantees that all neighbors of a source successfully receive and commit an application-level message from the source has been calculated. We have shown that this upper bound is only 0.081 seconds when the neighborhood span (2L) is 250 meters, the target group span is 625 meters, there are at most 26 neighbors in the neighborhood, there are at most 66 members in a group, the M-RBP token is passed every 0.001 second, and the maximum number of token/message recovery iterations allowed is 4. We have also shown that in an environment where the probability that a vehicle loses a message is less than or equal to 0.01, the probability that not all neighbors receive and commit an application-level message within 0.081 second is less than 10^{-7}. This shows that RNP is suitable for safety applications that have a strict message delivery delay requirement and rely on highly probable message delivery.

We have calculated the average number of messages that occur when an application-level message is transmitted, verified the results from the calculation with the simulations, and compared the results with a simple ARQ protocol that allows a source to repeatedly transmit an application-level message until it receives ACKs from all of its neighbors. The results show that the average number of messages occurring is only about 1 when RNP is used, while it is about N+1, where N is the number of neighbors, when the simple ARQ protocol is used. For example, when the number of neighbors (N) is 30, the probability that a vehicle loses a message is 0.01, and the message loss at all receiving

vehicles is fully correlated, the average number of messages transmitted by the simple ARQ protocol is 31.5791, while it is 1.0203 in RNP. On the other hand, when each receiving vehicle independently loses a message, the average number of messages is 31.7679 for the simple ARQ protocol and 1.5889 for RNP.

For RNP, the average number of messages transmitted is independent of the number of neighbors when there is no message loss or when the message loss at all receiving vehicles is fully correlated, while increases slightly with the number of neighbors when receiving vehicles independently lose a message. The increase in the latter case is due to extra NACKs from neighbors that miss the application-level message and application-level messages that are retransmitted to these neighbors. However, the number of messages transmitted by RNP increases at a much slower rate than the simple ARQ protocol. From these results, we can say that RNP is more efficient than the simple ARQ protocol in terms of the average number of messages transmitted per one application-level message and is more scalable with the number of neighbors. The scalability of RNP makes it suitable for VANETs in which node density can be high, such as on a congested highway.

An example of applying RNP in conjunction with sensors to avoid rear-end collisions has been presented in chapter 7. We have proposed a simple set of rules for using RNP in conjunction with sensors to automatically maintain a safe following distance, provide warnings of emergency situations, and negotiate the safe deceleration rates among nearby communicating vehicles in order to avoid rear-end collisions, and we have quantified the percentage of highway capacity improvement. In addition, we have

compared the highway capacity improvement from using RNP with sensors with that of using sensors alone. We have proposed simple rules to use sensors alone to automatically maintain a safe following distance to avoid rear-end collision.

Based on our proposed rules, using RNP with sensors can provide a very high percentage increase in highway capacity while helping avoid rear-end collisions. For example, in the case where all vehicles are equipped with sensors, use RNP for communicating with their neighbors, and move with the same constant speed, 100 km/h (62.14 mph), the highway capacity is about 3.74 times the capacity in the base case where all vehicles have neither sensors nor communication devices (273.67% increase in the highway capacity). This result points out that even with the simple set of rules, RNP has a potential of providing a high percentage of highway capacity improvement while simultaneously improving safety. Safety is improved because the rules always maintain the safe following distances, while human drivers may follow too closely.

Highway capacity improvement increases as the fraction of vehicles that are equipped with both sensors and RNP increases. The results show that highway capacity improves slowly when there is a small portion of equipped vehicles, then improves more and more quickly as the portion of equipped vehicles increases. This is because we have to communicate with nearby vehicles to benefit from communication protocols. Therefore, communication protocols will provide substantial benefits in highway capacity improvement when there is a large portion of vehicles equipped with communication devices.

Using RNP in conjunction with sensors provides a much higher percentage increase in highway capacity than using sensors alone. With the same 100 km/h speed assumption as previously mentioned, when all vehicles are equipped with sensors alone, the highway capacity is about 1.44 times the capacity in the base case (43.98% increase in highway capacity), which is much lower than the case of using RNP with sensors. This shows an advantage of using communication protocols with sensors over using sensors alone. Several automobile manufacturers are currently offering assisted driving systems that use sensors to automatically avoid collisions, automatic control systems based on communication protocols will be offered in the future as its value and advantages over sensors are demonstrated.

There are two reasons that communication protocols such as RNP can increase highway capacity and provide advantage over sensors. The first is from the small message delivery delays that the protocols can provide, which result in a shorter time for a communicating vehicle to detect that the communicating vehicle in front of it suddenly brakes, making it quicker than both the perception time of human drivers and the time it takes onboard sensors to sense and provide such information. The second reason is that communication protocols provide vehicles with information about vehicles beyond just their immediately preceding and following vehicles. This information can be used to further reduce the safe following distance and thus increase the highway capacity. For example, in our proposed rules, the information about deceleration rates of nearby communicating vehicles allow the vehicles to choose to use the same safe deceleration rate, so they do not need to maintain the part of the safe following distance due to differences in the deceleration rates.

A combination of an effective communication protocol and a novel application of the protocol is a key to significantly improving highway capacity and safety. Our reliable neighborcast protocol, RNP, is an effort to create a communication protocol that is suitable for the fast-changing environment of VANETs and provide a set of guarantees that are useful for VANET applications. The simple way that we have proposed to apply RNP to avoid rear-end collisions should provide a basic understanding of the percentage of highway capacity improvement that communication protocols could provide while improving safety and the advantage of using communication protocols in combination with sensors over using sensors alone. However, there is still a need for determining the best way to fully take advantage of RNP to increase highway capacity and improve safety. This includes determining what kind of information should be exchanged by neighboring vehicles and the best way to use the received information to control or coordinate the movements of vehicles to achieve the goals. An innovative way to apply RNP could result in even higher capacity improvement and higher level of safety.

References

[1] T. L. Willke, P. Tientrakool, and N. F. Maxemchuk, "A Survey of Inter-Vehicle Communication Protocols and their Applications," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 2, pp. 3-20, Jun. 2009.

[2] O. Wolfson and B. Xu, "Data-on-the-road in intelligent transportation systems," in *IEEE International Conference on Networking, Sensing, and Control (ICNSC 2004)*, (Taipei, Taiwan), 2004.

[3] L. Wischhof, A. Ebner, H. Rohling, "Information Dissemination in Self-Organizing Intervehicle Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 1, pp. 90-101, Mar. 2005.

[4] T. Kitani, T. Shinkawa, N. Shibata, K. Yasumoto, M. Ito and T. Higashino, "Efficient VANET-Based Traffic Information Sharing using Buses on Regular Routes," in IEEE Vehicular Technology Conference, 2008 (VTC Spring 2008), pp. 3031-3036, May 2008.

[5] T. Nadeem, S. Dashtinezhad and C. Liao, "Traffic view: A Scalable Traffic Monitoring System," in *Proceedings of the IEEE International Conference on Mobile Data Management (MDM 2004)*, pp.13–21, 2004.

[6] M. Saito, M. Funai, T. Umedu and T. Higashino, "Inter-vehicle Ad- hoc Communication Protocol for Acquiring Local Traffic Information," in *Proceedings of the 11th World Congress on Intelligent Transport Systems*, 2004.

[7] T. Delot , N. Cenerario , S. Ilarri , S. Lecomte, "A cooperative reservation protocol for parking spaces in vehicular ad hoc networks," in *Proceedings of the 6th International Conference on Mobile Technology, Application & Systems*, (Nice, France), pp. 1-8, Sep. 2009.

[8] S. B. Lee, G. Pan, J. S. Park, M. Gerla, and S. Lu, "Secure incentives for commercial ad dissemination in vehicular networks," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc07)*, (Montreal, Quebec, Canada), pp. 150-159, Sep. 2007.

[9] X. Yang, J. Liu, and F. Zhao, "A Vehicle-to-Vehicle Communication Protocol for Cooperative Collision Warning," in *Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)*, pp. 114-123, Aug. 2004.

[10] M. Durresi, A. Durresi, and L. Barolli, "Emergency broadcast protocol for inter vehicle communications," in *Proceedings of the 11th International Conference on Parallel and Distributed Systems Workshops (ICPADS'05)*, pp. 402-406, Jul. 2005.

[11] Q. Sun and H. Garcia-Molina, *Using Ad-Hoc Inter-Vehicle Networks for Regional Alerts*, Technical Report, Stanford Unversity, Stanford, California, USA, 2004.

[12] H. Alshaer and E. Horlait, "An optimized adaptive broadcast scheme for inter-vehicle communication," in *IEEE 61st Vehicular Technology Conference (VTC 2005-Spring)*, pp. 2840-2844, Jun. 2005.

[13] G. Korkmaz, E. Ekici, F. Ozguner, and U. Ozguner, "Urban Multi-Hop Broadcast Protocols for Inter-Vehicle Communication Systems," in *Proceedings of the First ACM Workshop VANET 2004*, pp. 76-85, Oct. 2004.

[14] M. Li and W. Lou, "Opportunistic broadcast of emergency messages in vehicular ad hoc networks with unreliable links," in *Proceedings of the 5th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine '08)*, (Hong Kong, China), Jul. 2008.

[15] R. Mangharam, R. Rajkumar, M. Hamilton, P. Mudalige, and F. Bai, "Bounded-Latency Alerts in Vehicular Networks", in *2007 Mobile Networking for Vehicular Environments*, pp. 55-60, May 2007.

[16] S. Oh, J. Kang, and M. Gruteser, "Location-based flooding techniques for vehicular emergency messaging," in *2006 Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*, pp. 1-9, Jul. 2006.

[17] J. F. Lee, C. Wang, and M. Chuang, "Fast and Reliable Emergency Message Dissemination Mechanism in Vehicular Ad Hoc Networks," in *Proceedings of IEEE Wireless Communications and Networking Conference(WCNC)*, pp.1-6, Apr. 2010.

[18] J. Peng and L. Cheng, "A Distributed MAC Scheme for Emergency Message Dissemination in Vehicular Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 6, pp. 3300-3308, 2007.

[19] D. Shin, H. Yoo, and D. Kim, "EMDOR: Emergency message dissemination with ACK-overhearing based retransmission," in First International Conference on Ubiquitous and Future Networks (ICUFN 2009), pp. 230-234, 2009.

[20] Y. Zhuang, J. Pan, Y, Luo, and L. Cai, "Time and Location Critical Emergency Message Dissemination for Vehicular Ad-Hoc Networks," IEEE Journal on Selected Areas in Communications, vol. 29, no. 1, pp. 187-196, Jan. 2011.

[21] J. VanderWerf, S. Shladover, M. Miller, and N. Kourjanskaia, "Evaluation of the effects of adaptive cruise control systems on highway traffic flow capacity and implications for deployment of future automated systems," *Pre-Print CD-ROM of 81st TRB Annual Meeting*, 2001.

[22] D. Ni, J. Li, S. Andrews, and H. Wang, "Preliminary Estimate of Highway Capacity Benefit Attainable with IntelliDrive Technologies," in *13th International IEEE Annual Conference on Intelligent Transportation Systems*, (Madeira Island, Portugal), Sep. 2010.

[23] T. W. Rand and M. S. Eby, "Algorithms for airborne conflict detection, prevention, and resolution," in *23rd Digital Avionics Systems Conference (DASC 04)*, pp. 3.B.1-1-3.B.1-17, 2004.

[24] A. Dogan et al., "Evaluation of intersection collision warning system using an inter-vehicle communication simulator," in *7th International IEEE Conference on Intelligent Transportation Systems*, (Washington, D.C.), pp. 1103-1108, Oct. 2004.

[25] R. Miller and Q. Huang, "An adaptive peer-to-peer collision warning system," in *IEEE Vehicle Technology Conference (VTC Spring 2002)*, (Birmingham, Alabama), pp. 317-321, 2002.

[26] Y. Liu and U. Ozguner, "Effect of inter-vehicle communication on rear-end collision avoidance," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp.168-173, Jun. 2003.

[27] D. Reichardt et al., "CarTALK 2000: safe and comfortable driving based upon inter-vehicle-communication," in *IEEE Intelligent Vehicle Symposium*, (Versailles, France), pp. 545-550, Jun. 2002.

[28] T. Tank and J.-P. Linnartz, "Vehicle-to-Vehicle Communications for AVCS Platooning", *IEEE Transactions in Vehicular Technology*, vol. 46, no. 2, pp. 528-236, May 1997.

[29] A. Hsu, F. Eskafi, S. Sachs, and P. Varaiya, "The Design of Platoon Maneuver Protocols for IVHS," PATH Research Report UCB-ITS-PRR-91-6, 1991.

[30] R. Tatchikou, S. Biswas, and F. Dion, "Cooperative vehicle collision avoidance using inter-vehicle packet forwarding," in *IEEE Global Telecommunications Conference (IEEE GLOBECOM '05)*, pp. 2762-2766, 2005.

[31] O. Gehring and H. Fritz, "Practical results of a longitudinal control concept for truck platooning with vehicle to vehicle communication," in *IEEE Conference on Intelligent Transportation System (ITSC 97)*, pp. 117-122, 1997.

[32] P. Seiler et al., "Disturbance propagation in vehicle strings," *IEEE Transactions on Automatic Control*, vol. 49, no. 10, pp. 1835-1841, Oct. 2004.

[33] S. Tsugawa et al., "A Cooperative driving system with automated vehicles and inter-vehicle communications in Demo 2000," in *Proceedings of the IEEE Intelligent Transportation Systems Conference*, (Oakland, CA), pp. 918- 923, Aug. 2001.

[34] L. Li, F-Y. Wang, "Cooperative Driving at Blind Crossings Using Intervehicle Communication," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 6, pp. 1712-1724, Nov. 2006.

[35] S. Matsuda et al., "Vehicular information broadcasting relay (VIBROR) protocol for inter-vehicle-communications," in *52nd IEEE Vehicular Technology Conference (VTS-Fall 2000)*, (Boston, MA), pp. 2005-2010, 2000.

[36] M.-T. Sun et al., "GPS-based message broadcasting for inter-vehicle communication," in *International Conference on Parallel Processing*, (Toronto, Ontario, Canada), pp. 279-286, 2000.

[37] Z. Zhang, "Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges," *IEEE Communications Surveys Tutorials*, vol. 8, no.1, pp. 24-37, 2006.

[38] H. Wu et al., "MDDV: a mobility-centric data dissemination algorithm for vehicular networks," in *1st ACM International Workshop on Vehicular Ad Hoc Networks (VANET'04)*, (Philadelphia, PA), pp. 47-56, 2004.

[39] W. Chen and S. Cai, "Ad hoc peer-to-peer network architecture for vehicle safety communications," *IEEE Communications Magazine*, vol. 43, pp. 100-107, Apr. 2005.

[40] H.-J. Reumerman et al., "The application-based clustering concept and requirements for intervehicle networks," *IEEE Communications Magazine*, vol. 43, pp. 108-113, Apr. 2005.

[41] D. Lee et al., "A Wireless Token Ring Protocol for intelligent transportation systems," in *Proceedings of 2001 IEEE Intelligent Transportation Systems Conference*, (Oakland, California), pp. 1152-1157, Aug. 2001.

[42] T. L. Willke and N. F. Maxemchuk, "Reliable collaborative decision making in mobile ad hoc networks," in *Proceedings of 7th IFIP/IEEE International Conference MMNS*, (San Diego, CA), pp. 88–101, Oct. 3–6, 2004.

[43] T. L. Willke and N. F. Maxemchuk, "Coordinated interaction using reliable broadcast in mobile wireless networks," *The International Journal of Computer and Telecommunications*, vol. 51, Issue 4, Mar. 2007.

[44] ASTM Standard E2213, 2003 (2010), "Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle Systems-5 GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications," ASTM International, www.astm.org.

[45] J. D. Hill, G. Rhodes, S. Voller, and C. Whapples, *Car Park Designers' Handbook.* Thomas Telford Limited, London, 2005.

[46] D. B. Maciuca and K. J. Hedrick, "Brake Dynamics Effect on AHS Lane Capacity," in *Future Transportation Technology Conference & Exposition*, Aug. 1995.

[47] R. Bishop, "A survey of intelligent vehicle applications worldwide," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, (Dearborn, Michigan), Oct. 2000.

[48] T. Chang and I. Lai, "Analysis of characteristics of mixed traffic flow of autopilot vehicles and manual vehicles," *Transportation Research Part C*, vol. 5, no. 6, pp. 333–348, 1997.

[49] A. Mehmood and S. M. Easa, "Modeling Reaction Time in Car-Following Behaviour Based on Human Factors," *International Journal of Applied Science, Engineering and Technology*, vol.5, no. 2, pp. 93–101, 2009.

[50] VanderWerf, J. et al., "Modeling the Effects of Driver Control Assistance Systems on Traffic", in *Proceedings of the 80th Annual Meeting of the Transportation Research Board*, (Washington D.C.), Jan. 2001.

[51] Transportation Research Board, *Highway Capacity Manual.* National Research Council, Washington, DC, 2000.