ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/313435212

NS-2 Simulation of VANET for Safety Applications: Issues and Solutions

Conference Paper · January 2017

DOI: 10.1145/3036331.3036349

CITATIONS
0

READS
47

5 authors, including:

Jingyu Li
Harbin Engineering University
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Jing Zhao
Harbin Engineering University
33 PUBLICATIONS **264** CITATIONS
SEE PROFILE

Xiaomin Ma
Oral Roberts University
92 PUBLICATIONS **1,086** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Analytic Modeling and Enhancement of Vehicular Ad Hoc Networks for Safety Critical Applications View project

Software aging and rejuvenation View project

All content following this page was uploaded by Xiaomin Ma on 22 March 2017.

The user has requested enhancement of the downloaded file.

NS-2 Simulation of VANET for Safety Applications: Issues and Solutions

Jingyu Li
Computer Science and Tech. Dept.
Harbin Engineering University
Harbin, 150001, China
lijingyu@hrbeu.edu.cn

Yunan Zhang
Computer Science and Tech. Dept.
Harbin Engineering University
Harbin, 150001, China
zyn1949@163.com

Jing Zhao
Computer Science and Tech. Dept.
Harbin Engineering University
Harbin, 150001, China
jingzhao.duke@gmail.com

Yanbin Wang
Department of Industrial Engineering
Harbin Institute of Technology
Harbin, 150001, China
wangyb@hit.edu.cn

Xiaomin Ma
College of Science & Engineering
Oral Roberts University
Tulsa, OK, 74171, USA
xma@oru.edu

Wei Wu
The Fifty Fourth Institute of CETC
Shijiazhuang, 050081, China

ABSTRACT

NS-2, as an open-source network simulator, is widely used in wireless communication studies and VANET safety applications. In [1], Chen et al. implemented a completely modified architecture about IEEE 802.11 MAC and physical layer modules which depicted the signal transmission and reception process in IEEE 802.11, as well as the method of handling collision. However, in the process of using the latest version of NS2, we found some problems. First, its approach of deciding whether to receive the signal by comparing the *SINR* and threshold is not flexible in actual simulation. Second, the capture thresholds for collision do not work well. Third, as for the *RXThreshold_* property, class *WirelessPhyExt* doesn't achieve relevant functions. In this paper, we put forward a viable scheme to solve these problems by deeply understanding the signal reception process based on SINR.

CCS Concepts

• **Security and privacy → Mobile and wireless security**
• **Networks → Network simulations**

Keywords

IEEE 802.11; NS2; Simulator; VANET

1. INTRODUCTION

In the current study of vehicle ad-hoc network, the method combining theory with simulation is often used in the theoretical analysis of safety applications to verify each other. In [2], [3], for example, theoretical values calculated by Matlab and simulation results of NS2 were compared with each other to demonstrate the conclusions. NS2, as an open-source network simulator, is utilized frequently to simulate VANET safety applications. Therefore, it's very necessary to understand the internal modules about IEEE 802.11 in NS2, which can help us to simulate effectively.

In NS2 code, {*Mac802_11, WirelessPhy*} and {*Mac802_11Ext, WirelessPhyExt*} are all based on IEEE 802.11. Herein, *Mac802 11* and *Mac802 11Ext* are implemented in the MAC layer, while *WirelessPhy* and *WirelessPhyExt* are in the physical layer. In *WirelessPhy*, when the signal power is greater than *RXThreshold_*, the signal is acceptable. And the signal will be received when the SINR (Signal to Interference plus Noise Ratio) of the signal is more than the demodulation threshold in *WirelessPhyExt*. Thus, {*Mac802_11, WirelessPhy*} and {*Mac802_11Ext, WirelessPhyExt*} are significantly different.

In [1], [4], Chen et al. detailed the signal transmission and reception process in IEEE 802.11, as well as two kinds of collision. Then the literatures depicted the implementation for the physical layer and the MAC layer of IEEE 802.11 within NS2, and pointed out that it judges whether to accept the signals via SINR in collision. Hence, what Chen et al. said refer to *Mac802_ 11Ext* and *WirelessPhyExt*. Meanwhile, Chen et al. are also the developers of *WirelessPhyExt*.

In IEEE 802.11 of NS2 described in the literatures [1], [4], there are two modules placed in the physical layer, which are the power monitor and the state manager. And the state manager is responsible for collision management and introduces four operating modes which are *TXing* state for transmitting a frame, *Searching* state for waiting for a frame, *Pre_RXing* state for receiving preamble and *RXing* state for receiving body of a frame.

When a transmit command comes from the MAC layer, the physical layer moves into *TXing* state for the duration of a frame transmission regardless what the physical layer is doing at that moment. Within *Searching* state, the physical layer is neither in transmission nor reception of a frame. The physical layer stays in the *Pre_RXing* state for the process of signal preamble. *RXing* state refers to the physical layer processes the reception of the body of the current frame.

Recently, however, in the process of using the latest version of NS2, we found some problems. The version we are using is v2.35, and the classes are *Mac802_11Ext*, *WirelessPhyExt* and *Agent*. The issues are as follows:

1. In the class of *WirelessPhyExt*, it decides whether to receive the signal by comparing the SINR and threshold value. However this method is not flexible in actual simulation. Because it mainly sets threshold value by changing the *BasicModulationScheme* property, which is feasible, but the changing range of threshold has only four values: 3.1623,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
ICCMS '17, January 20-23, 2017, Canberra, Australia
© 2017 ACM. ISBN 978-1-4503-4816-4/17/01…$15.00
DOI: http://dx.doi.org/10.1145/3036331.3036349

6.3096, 31.6228 and 316.2278. Therefore, this approach is not very convenient for the simulation.

2. In the collision of two packets from the channel, the capture of signal by the *SINR* is not effective as literature [1] guaranteed for what the capture switch is closed or not doesn't have much impact for the *PRR* (Packet Reception Ratio).

3. As for the *RXThreshold_* property, the class of *WirelessPhyExt* does not carry out relevant functions, so it can't change the signal transmission distance by setting the value of *RXThreshold_*.

The aforementioned problems are what we found during actual simulation and our main work is putting forward a viable scheme to solve these problems by deeply understanding the signal reception process based on SINR.

2. UNDERSTANDING AND ANALYSIS OF NS2 SINR-BASED PACKET RECEPTION

2.1 Basic idea and process

In NS2, *WirelessPhyExt* inherits and overrides two methods of *SendUp()* and *SendDown()* where *SendUp()* tackles packets from the channel and *SendDown()* handles packets from the MAC layer.

When a packet comes from the channel, *SendUp()* will drop this packet or pass it to the MAC layer. As shown in Figure 1, when *SendUp()* receives a packet from the channel, it first calculates the signal power P_r which is determined by the transmit power of signal the fading channel model. After the packet arrives, *SendUp()* calls the method of *recordPowerLevel()* of *PowerMonitor*, and *recordPowerLevel()* will rerecord the latest signal power of the current channel. Then the received signal power denoted by *signalPower* is compared with the monitor threshold *monitor_Thresh*. If greater than *monitor Thresh*, *signalPower* will be added to the current signal power and the sum is expressed as *powerLevel*. Otherwise, the signal will be ignored. After recording signal, the physical layer moves into the corresponding state to process the signal

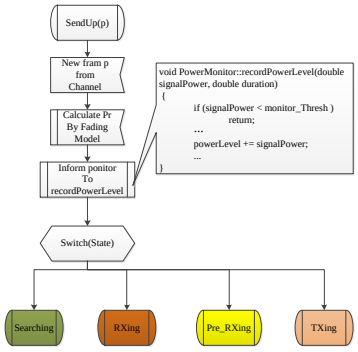

Figure 1. *SendUp()* Method

Assuming that when the signal S_1 arrives from channel, the physical layer is in *TXing* state, then the signal will just be ignored like the Figure 2. If under *Searching* state when the signal S_1 coming from the Channel arrives, the physical layer will deal with this signal according to Figure 2. Firstly, the physical layer compares *SINR* of S_1 with the demodulation threshold which has only four values available: 3.1623, 6.3096, 31.6228 and 316.2278. If $SINR(S_1)$ is higher than or equal to the demodulation threshold, the physical layer moves into *Pre_RXing* state continuing to process S_1 as shown in Figure 3. At this moment, the signal collision may occur, and *WirelessPhyExt* allows to capture the signal. Otherwise, the signal will be ignored directly and the physical layer stays in *Searching* state.

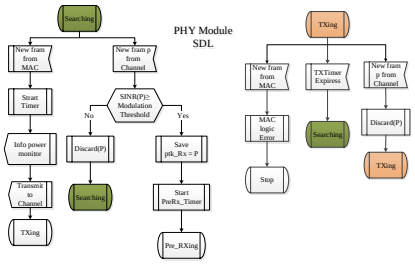

Figure 2. *Searching* and *TXing* States of PHY Module SDL

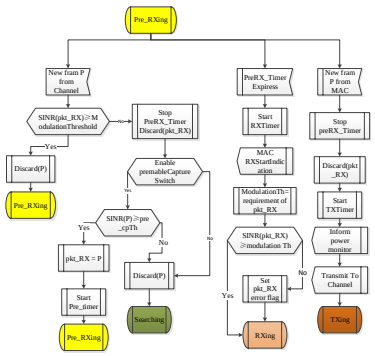

Figure 3. *Pre_RXing* State of PHY Module SDL

Herein, *SINR* of S_1 is calculated as follows:

$$SINR(S_1) = \frac{P_r(S_1)}{getPowerLevel() - P_r(S_1)}$$

In the above formulation, $P_r(S_1)$ refers to the power of S_2 after received, *getPowerLevel()*, one of the methods of *PowerMonitor*, are able to obtain the current signal power *powerLevel*.

If the channel is free, *getPowerLevel()* will return a free constant value *noise_floor*, or return *powerLevel*. Assuming that there are two signals S_1 as well as S_1, where S_1 arrives first. According to aforementioned formula, it is found when the signal collision occurs, *SINR* of S_1 can be calculated as $SINR(S_1) = P_r(S_1)/P_r(S_2)$. Because when S_2 arrives, *recordPowerLevel()* in *PowerMonitor* has been called, which adds S_2 to the current channel signal power, so the signal power returned by *getPowerLevel()* is equal to the sum of signal power of S_1 and S_2.

As shown in Figure 3, when S_2 arrives as a new signal, the physical layer is under *Pre_RXing* state, then it will regard S_2 as noise, and compare $SINR(S_1)$ with the demodulation threshold. If $SINR(S_1)$ is higher, the physical layer is going to receive S_1, and S_2 will be ignored directly. Otherwise, the physical layer ignores S_1 and discards being received data to view the switch of *premableCaptureSwitch* is turned on or not. If it's on, S_1 will be regarded as noise, and $SINR(S_2)$ is compared with capture threshold *SINR preambleCapture*. If the former is greater, the physical layer is ready to receive S_2 and remains in *Pre_RXing* state, or abandons S_2. If the switch of *premableCaptureSwitch* is off, the physical layer ignores S_2 and turns to *Searching* state.

When the physical layer is under *RXing* state of S_1 and S_2 arrives as a new signal, the physical layer will handle signals in terms of Figure 4. Firstly, it treats S_2 as noise, and makes a comparison between $SINR(S_1)$ and the demodulation threshold. If $SINR(S_1)$ is greater than the demodulation threshold, the physical layer ignores S_2 and continues to receive S_1, or marks S_1 an error flag. At the

same time, the physical layer judges whether the capture switch is open. If the switch is on, we regard S_1 as noise, then compares $SINR(S_2)$ with the capture threshold *SINR_DataCapture_*. If $SINR(S_2)$ is higher, the physical layer will neglect S_2 and abandon the data being read getting ready to receive the S_2. At the same time, the physical layer switches into *Pre_RXing* state. If the switch is off, the physical layer ignores S_2 directly and remains in *RXing* state.

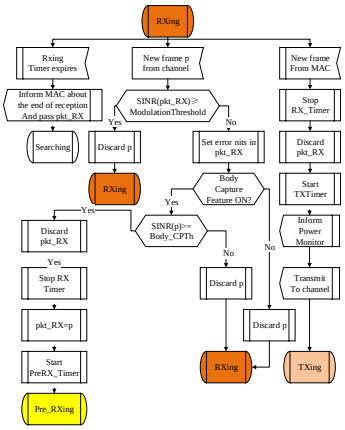

Figure 4. *RXing* State of PHY Module SDL

All above is the reception process of *WirelessPhyExt*, including the normal reception and collision capture. But when use the capture function in the simulation, we can barely capture the collision packet and found that the captured packets have little effect on the *PRR*. In the following section, we will analyze the reasons about this phenomenon.

2.2 Explain why the capture thresholds do not work well (from the flow diagram)

It is known from the above description that the signal collision occurs in *Pre_RXing* state and *RXing*.

Within *Pre_RXing* state, after fails to demodulate the first signal and *premableCaptureSwitch* is on, the physical layer starts to capture the second signal, i.e. the new signal. As shown in Figure 3, when *SINR* of the second signal is equal to or greater than *SINR preambleCapture*, the physical layer tries to receive it. After reading preamble of the signal, the physical layer can get the modulation mode of current signal, and modulation threshold is the same with modulation threshold, that is, 3.1623, 6.3096, 31.6228 and 316.2278. Later, *SINR* of the second signal is compare with the demodulation threshold. If *SINR* is lower, the signal will be marked error flag and ignored. Otherwise, the physical layer receives the data part of signal and moves to *RXing* state. Since both the capture threshold and the modulation threshold are compared with *SINR*, it is necessary to take into account the relationship of them, that is, whether the capture threshold should be greater than the modulation threshold or not.

Unable to receive the first signal, the physical layer wants to read the second one, and enable the capture function. As Figure 3 shows, under the circumstances where the capture threshold is less than the modulation threshold, if *SINR* is lower than the capture threshold, it fails to capture the signal, and if *SINR* is greater than capture threshold, but less than the modulation threshold, the signal will also be ignored and captured unsuccessfully. Thus, it is observed that the size of the capture threshold does not affect the capture of signals, because *SINR* of the signal will eventually be compared to the modulation threshold.

On the contrary, if the capture threshold is greater than or equal to the modulation threshold, the physical layer must be able to enter *RXing* state when the second signal is captured successfully, so it is not meaningful to compare *SINR* of the signal with the modulation threshold at the end of the *pre_Timer* timer. Meanwhile, the condition that *SINR* is greater than the capture threshold is too harsh makes a lower capture probability.

When the collision occurs in *RXing* state and the capture switch is on, the physical layer makes a comparison between *SINR* of the captured signal and *SINR_DataCapture_*. If *SINR* is greater than or equal to *SINR_DataCapture_*, the physical layer tries to receive signals and switches into *Pre_RXing* state. At the end of the *pre_Timer*, as shown in Figure 3, SINR of the captured signal and the modulation threshold will be compared with each other.

Table 1 shows the percentage of packets captured successfully in *Pre_RXing* state and *RXing* state after enabling the capture function under different modulation and demodulation modes.

Table 1. The Probability of Capturing Signals Successfully Under Different Modulation Scheme

Modulation Scheme	*Pre_RXing* State	*RXing* State
0	37.5%	15%
1	31%	15.6%
2	19%	14.2%
3	6.88%	3.82%

It's known that the probability of capturing signals successfully becomes lower as modulation scheme changes from 0 to 3. Because the capture threshold is higher with greater modulation scheme, which verifies the growth of capture threshold will degrade the capture rate.

Table 2 is the NS2 parameters we used.

Table 2. NS2 Parameters

HeaderDuration_	44e-6	CSThresh_	2.39455e-11
SymbolDuration_	4e-6	PowerMonitorThresh_	2.39455e-11
SlotTime_	1.6e-5	L_	1
SIFS	3.2e-5	SINR_preambleCapture_	2.5118
ShortRetryLimit_ LongRetryLimit_	1000	SINR_DataCapture_	100
RTSThreshold_	10000	freq_	5.18e+9
Agent	PBC	bandwidth_	10e6
periodicBroadcastInterval	0.1	Pt_	0.28183815
Channel: Nakagami	3: $x<50$;1.5: $50\le x<150$;1: $x<150$		

[1] x means the distance between two communication nodes

As seen in Table 3, the first column indicates the modulation scheme, the second column respects whether the capture function is enabled, the third to sixth columns depict the amount of drop packets under *Pre_RXing* state and *RXing* state when in collision and the last column shows the total number of lost packets in the experiments.

when the modulation scheme is 0 (*BPSK*), it can be found the number of dropped packets due to collision accounts for 50% of the total number, which is a very high rate, thus, we enable the capture function of *WirelessPhyExt*, and as expected the packet loss events by collision should be reduced.

Now we represent the experimental procedures. The *modulationScheme*, which is set when the *PBC* sends packets, is the same as *BasicModulationScheme_* configured by the physical layer of receiver, namely, transmitting and receiving packets have the same modulation-demodulation method. Communication distance is 500 meters which is the same with sensing distance. However, the setting of communication distance and sensing distance in *WirelessPhyExt* still exists some problems, we will specify them in the next section. In the experimental, the number of nodes is 1000, and they are evenly distributed over the one-dimensional line. In the different modulation modes, that is, 0, 1, 2, 3, we get a set of data below (see Table 3).

Table 3. Packet Loss under *RXing* and *Pre_RXing* States

Modulation Scheme	Capture Switch (preamble and data)	the amount of drop packets under collision				The total drop packets
		Pre_RXing Period		*RXing* Period		*Rxing Pre_RXing Searching Txing*
		1st	2nd	1st	2nd	
0	OFF	34988	314321	4.28831e+06	702952	10784915
	ON	33579	313491	4.43016e+06	581142	10705679
1	OFF	31449	310022	2.64112e+06	344781	10483538
	ON	36164	307067	2.75174e+06	329332	10400857
2	OFF	33125	288717	1.63442e+06	191261	10340207
	ON	35271	295262	1.68058e+06	188904	10285867
3	OFF	30007	279220	1.00944e+06	106043	10309605
	ON	37237	295331	1.05673e+06	118319	10236443

From results shown in Table 3, however, it's known that enabling the capture function does not have much impact for packet loss events. And Figure 5 also shows that *PRR*, in different modulation scheme, has scarcely reaction for the status of the capture switch.

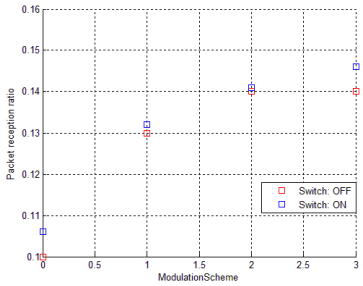

Figure 5. Simulation results for the network

2.3 About other thresholds: RX thresholds, carrier sensing threshold, and monitor power threshold (connection and differences)

Generally in the simulation, we are concerned much more about the communication distance and sensing distance, therefore, he Mac-802.11 protocol in NS2 provides several related properties for us, such as P_t, *Channel*, *RXThreshold_* and *CSThreshold_*. To the best knowledge, if P_t and *Channel* are fixed, we can change the communication distance and sensing distance by setting *RXThreshold_* and *CSThreshold_* respectively. Meanwhile, *RXThreshold_* and *CSThreshold_* are all in power units. However, in *WirelessPhyExt* we introduces the concept of SINR to receive signal, so the setting of *RXThreshold_* and *CSThreshold_* will be a little different.

In *WirelessPhyExt* code, we know *RXThreshold_* did not attain relevant functions, and only defined the variables in *tcl* script, which is also found in the *WirelessPhyExt* flow diagrams (shown as Figure 2, Figure 3 and Figure 4).

The *CSThreshold_*, in *WirelessPhyExt*, is still utilized to monitor the channel. When the signal power is below *CSThreshold_*, we think that the current channel is occupied related to which is *recordPowerLevel()* of *PowerMonitor* module. When a signal arrives, it calls this method to record the latest signal power of current channel. As shown in Figure 1, in *recordPowerLevel()*, the signal power of new arrival signal will be compared with *monitor Thresh*. If the signal power is greater than or equal to the threshold, the signal will be cumulated to the channel signal power, or it will be ignored. And *monitor_Thresh* is corresponded to the *PowerMonitorThresh_* property of *WirelessPhyExt*. As we all know, only perceiving the signal firstly can the physical layer receive it. Thus, it's able to sense the signal as long as this signal produces an effect on the physical layer. Therefore, if the signal is cumulated to the current channel signal power, the physical layer should be able to sense the signal. So the value of *PowerMonitorThresh_* should be equal to or higher than *CSThreshold_* In the experiment, we deem that the signal can affect the physical layer because of being perceived. Thus, in the simulation, the *PowerMonitorThresh_* is the same with *CSThreshold_*.

3. NEW SCHEMES TO IMPLEMENT NS2 SINR-BASED PACKET RECEPTION

For the situation that *RXThreshold_* does not achieve relevant functions, as well as *modulationThreshold* has finite values, we need to revise *WirelessPhyExt*. The following are a few kinds of feasible solutions.

3.1 Make change from **.h* files

If we aren't very familiar with the mechanism of NS2 but have the emergency need, it's recommended to adopt this method. For example, we need to set different *RXThreshold_* values, while *MoudulationThreshold* has only four values. At this moment, we can modify the *modulation_table* array in *WirelessPhyEx.h*, the program fragment are as follows. Suppose the demodulation method we use is BPSK, and the required threshold is 1, we just need to change 3.1623 in *BPSK* array into 1, and then recompile the NS2 program.

```
const struct ModulationParam modulation table[4] ={
        // mod name SINRdB SINR NDBPS bit
        { 0 , "BPSK", 5 , 3.1623 , 24 } ,
        { 1 , "QPSK", 8 , 6.3096 , 48 } ,
        { 2 , "QAM16", 15 , 31.6228 , 96 } ,
        { 3 , "QAM64", 25 , 316.2278 , 192 }
};
```

While if we need to change thresholds frequently, the simulation will be very complex using above methods. Thus, we can replace

modulation_table[BasicModulationScheme_].SINR_ratio and *SINR_Th_RX* of *SendUp()*, as well as *SINR_Th_RX* of *handle_PreRXtimeout()* with *RXThreshold_* , so that we can set transmission distance by *RXThreshold()* in *tcl* script.

3.2 Make change from **.tcl* files

If we need more changes, for example, the capture way in *WirelessPhyExt.cc* is not what we want, we can maintain a copy of *WirelessPhyExt{.h .cc}* and *Mac-802Ext{.h .cc}* and rename them as *MyWirelessPhyExt{.h .cc}* and *myMac-802Ext{.h .cc}* respectively.

1. Add all methods and properties of *MyWirelessPhyExt{.h .cc}* and *MyMac-802.Ext{.h .cc}* into a namespace, meanwhile change the name of *include* file and macro definition, otherwise there will be error.
2. Replace *HDR_MAC802_11* structure name in *MyMac-802.Ext.h* with self-defined name. In the first 58 lines of *packet.h*, there is the macro definition about *HDR_MAC802_11*, so we can follow this format to define our own macro definitions. Then replace all *HDR_MAC802_11* in *MyWirelessPhyExt{.h .cc}* and *MyMac-802.Ext{.h .cc}* with our own macro definitions.
3. There is a function bound to *tcl* script in *MyWirelessPhyExt.cc* and *MyMac-802.Ext.cc* respectively, which are *WirelessPhyExt* Class and *Mac802_11Ext* Class. Therefore, we need to change their names, such as *TclClass(Mac/802_11Ext)* can be changed into *TclClass(Mac/My802_11Ext)*, then we can declare properties as *Mac/My802_11Ext set CSThreshold_ 2.39455e-11* in the tcl script [5], [6]. Remember to replace *return(new Mac802_11Ext())* to *return(newMyMac802_11Ext())*, of which *MyMac802_11()* is our own definition of the class name to substitute the *Mac802_11Ext*.
4. Modify *calcHighestAntennaZ()* method in *Channel.cc*, which is used to calculate the gain. We model *WirelessPhyExt* to add custom judgment in *MyWirelessPhyExt*.
5. Change the *Makefile* file and **.tcl* files of NS2. Here we can refer to the literatures [5], [6], [7] and website [8].

3.3 Numerical results to show effectiveness of the new schemes

Here we use the second method to do simulation in which we replace demodulation threshold with *RXThreshold_*. But *RXThreshold_* is in the ratio instead of in watts. In order to reflect the effect, *RXThreshold_* of *MyWirelessPhyExt* is set to 31.6228, while the *BasicModulationScheme* of *WirelessPhyExt* and the *modulationScheme* of agents *PBC* are set into 3, namely, the demodulation threshold is 31.6228. Here are a group of data of loss packets shown in Table 4.

Table 4. Packet Loss in *WirelessPhyExt* & *MyWirelessPhyExt*

WirelessPhyExt				*MyWirelessPhyExt*			
Collision: the number of dropped packets							
Pre_RXing		*RXing*		*Pre_RXing*		*RXing*	
1st	2nd	1st	2nd	1st	2nd	1st	2nd
44	1959	11695	899	25	1935	835	11795
Total dropped packets in four states							
394796				401046			

Based on experimental data, it is proved that revised *MyWirelessPhyExt* can be used normally. Here we just briefly introduce how to add your own modification protocol, and if need to add the agent, you can refer to the NS2 manual in which there are more details, here we will not repeat.

4. CONCLUSION

From the flow diagrams and source code of *WirelessPhyExt*, as well as experimental results, we conclude that capture function in *WirelessPhyExt* is not very effective and falls flat. Secondly, when setting the sensing distance, *CSThreshold_* should be the same with *PowerMonitorThresh_*. And *CSThreshold_* can affect the transmission distance through energy cumulation for which nodes out of the transmission distance may also receive broadcast packets.

Against the defects in *WirelessPhyExt*, we offer two modification methods by which readers can revise source code of *WirelessPhyExt* according to their own needs. Of course, we can also combine two approaches defining our own protocols by inheritance.

In simulation of VANET for safety applications, the transmission of security information is generally in form of broadcast. When the sensing distance and communication distance are the same, the hidden terminal will lead to collision which has a high probability. Therefore, it's necessary to investigate this kind of collision events. In this article, we provide two ways to modify *WirelessPhyExt*, which is with the guidance to make more improvements on *WirelessPhyExt* in the future simulation.

5. ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Award No. 61572150).

6. REFERENCES

[1] Q. Chen, F. Schmidt-Eisenlohr, D. Jiang, M. Torrent-Moreno, L. Delgrossi, and H. Hartenstein, "Overhaul of ieee 802.11 modeling and simulation in NS2," in Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems. ACM, 2007, pp. 159–168.

[2] X. Ma and K. S. Trivedi, "Reliability and performance of general two-dimensional broadcast wireless network," *Performance Evaluation*, vol. 95, pp. 41–59, 2016.

[3] X. Ma, J. Zhang, and T. Wu, "Reliability analysis of one-hop safety-critical broadcast services in vanets," IEEE transactions on vehicular technology, vol. 60, no. 8, pp. 3933–3946, 2011.

[4] D.-C. README, "Overhaul of ieee 802.11 modeling and simulation in NS2 (802.11 ext)," Available at: dsn. tm. uka. de/medien/.../Documentation-NS2-80211Ext-2008-02-22. pdf

[5] K. Liu, "Understanding the implementation of ieee mac 802.11 standard in NS2," The VINT Project: A Collaboration between Researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC, 2007.

[6] T. Issariyakul and E. Hossain, "Introduction to network simulator ns2 manual," 2009.

[7] Y.-S. Yang, "Cloning a new ieee 802.11 mac protocol in ns2."

[8] http://blog.sina.com.cn/s/blog 69e6c0030100yq0x.html.

Columns on Last Page Should Be Made As Close As Possible to Equal Length

Authors' background

Your Name	Title*	Research Field	Personal website
Jingyu Li	master student	Wireless Network Distributed System	
Yunan Zhang	master student	Wireless Network	
Jing Zhao	full professor	reliability engineering software aging theory dependability modeling computer network	zhaojing.hrbeu.edu.cn
Yanbin Wang	associate professor	quality management, scheduling, and optimization	
Xiaomin Ma	full professor	computer and communication networking, computational intelligence	http://www.oru.edu/academics/faculty-profiles/profile.php?id=163
Wei Wu	senior lecture	Communication Network	

*This form helps us to understand your paper better, the form itself will not be published.

*Title can be chosen from: master student, Phd candidate, assistant professor, lecture, senior lecture, associate professor, full professor

View publication stats