International Research Journal of Advanced Engineering and Science

ISSN: 2455-9024

Worm Hole Attack Detection in Wireless Sensor Network

G. Elumalai[1], U. Vasudevan[2]

[1, 2]Panimalar Engineering College, Chennnai-600123

Email address: [2]u.vasudevan85@gmail.com

Abstract— *Network coding has been shown to be an effective approach to improve the wireless system performance. In network coding systems, the impact of wormhole attacks and counter measures are unknown. In this paper, we quantify wormholes 'attack impact on network coding system performance through experiments. We first propose a centralized algorithm to detect wormholes and show its correctness rigorously. For the distributed wireless network, we propose DAWN, a Distributed detection Algorithm against Wormhole, by exploring the change of the flow directions of the innovative packets caused by wormholes. We rigorously prove that DAWN guarantees a good lower bound of successful detection rate. We perform analysis on the resistance of DAWN against collusion attacks. Extensive experimental results have verified the effectiveness and the efficiency of DAWN.*

Keywords— *WSN, RLNC system, network simulator.*

I. Introduction

The main aim of this project is how to detect a wormhole attack using Expected Transmission count technique in Random linear Network coding (RLNC) System .Wormhole link will be removed in the network and packets are transmitted through secure link To improve the system performance of wireless Network, network coding is shown to be effective approach and it is totally different from traditional network In contrast, in wireless network coding systems, the forwarders are allowed to apply encoding schemes on what they receive, and thus they create and transmit new packets.

To improve the system performance of wireless networks, network coding has been shown to be an effective and promising approach (e.g., [I], [II]) and it constitutes a fundamentally different approach compared to traditional networks, where intermediate nodes store and forward packets as the original. In contrast, in wireless network coding systems, the forwarders are allowed to apply encoding schemes on what they receive, and thus they create and transmit new packets. The idea of mixing packets on each node takes good advantages of the opportunity diversity and broadcast nature of wireless communications, and significantly enhances system performance. However, practical wireless network coding systems face new challenges and attacks, whose impact and countermeasures are still not well understood because their underlying characteristics are different from well-studied traditional wireless networks. The wormhole attack is one of these attacks. In a wormhole attack, the attacker can forward each packet using wormhole links and without modifies the packet transmission by routing it to an unauthorized remote node. Hence, receiving the rebroadcast packets by the attackers, some nodes will have the illusion that they are close to the attacker.

II. System Analysis

A. Existing System

In traditional network, use connectivity graphs with binary relation (i.e., connected or not) on the set of nodes. For this reason, prior works based on graph analysis cannot be applied. Introduced by wormhole attacks to detect them unfortunately, this type of solutions cannot work with network coding either. They require either to use an established route that does not exist with network coding, or to calculate the delay between every two neighboring nodes which will introduce a huge amount of error in network coding systems.

B. Proposed System

In this project we detect a wormhole attack. Packet will be sending using RLNC technique, when the attacker attack the nodes while sending data. We find Critical Nodes and then we remove that path. Packet will be sending other paths and expected transmission count is to be increased if wormhole attack is detected.

We first propose a centralized algorithm to detect wormholes leveraging a central node in the network. For the distributed scenarios, we propose a distributed algorithm, DAWN, to detect wormhole attacks in wireless intra-flow network coding systems. The main idea of our solutions is that we examine the order of the nodes to receive the innovative packets in the network. Innovative Packet means intermediate node receives a packet which is linearly independent from previous packets.

III. Technologies Used

A. NS (version 2)

NS (version 2) is an object-oriented, discrete event driven network simulator developed at UC Berkely written in C++ and OTcl. NS is primarily useful for simulating local and wide area networks. Although NS is fairly easy to use once you get to know the simulator, it is quite difficult for a first time user, because there are few user-friendly manuals. Even though there is a lot of documentation written by the developers which has in depth explanation of the simulator, it is written with the depth of a skilled NS user. The purpose of this project is to give a new user some basic idea of how the simultor works, how to setup simulation networks, where to look for further information about network components in simulator codes, how to create new network components, etc., mainly by giving simple examples and brief explanations based on our experiences. Although all the usage of the simulator or possible network simulation setups may not be covered in this project, the project should help a new user to get started quickly.

NS is an event driven network simulator developed at UC Berkeley that simulates variety of IP networks. It implements network protocols such as TCP and UPD, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms such as dijkstra's algorithm, and more. NS also implements multicasting and some of the MAC layer protocols for LAN simulations. The NS project is now a part of the VINT project that develops tools for simulation results display, analysis and converters that convert network topologies generated by well-known generators to NS formats. Currently, NS (version 2) written in C++ and OTcl (Tcl script language with Object-oriented extensions developed at MIT) is available

Most of the figures that are used in describing the NS basic structure and network components are from the 5th VINT/NS Simulator Tutorial/Workshop slides and the NS Manual (formerly called "NS Notes and Documentation"), modified little bit as needed. For more information about NS and the related tools, visit the VINT project home page.

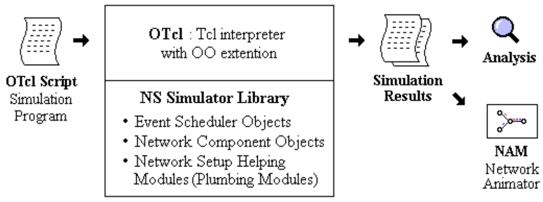

OTcl Script
Simulation Program
OTcl : Tcl interpreter with OO extention
NS Simulator Library
• Event Scheduler Objects
• Network Component Objects
• Network Setup Helping Modules (Plumbing Modules)
Simulation Results
Analysis
NAM
Network Animator

Fig. 1

As shown in figure 1, in a simplified user's view, NS is Object- oriented Tcl (OTcl) script interpreter that has a simulation event scheduler and network component object libraries, and network setup (plumbing) module libraries (actually, plumbing modules are implemented as member functions of the base simulator object). In other words, to use NS, you program in OTcl script language. To setup and run a simulation network, a user should write an OTcl script that initiates an event scheduler, sets up the network topology using the network objects and the plumbing functions in the library, and tells traffic sources when to start and stop transmitting packets through the event scheduler.

NS is written not only in OTcl but in C++ also. For efficiency reason, NS separates the data path implementation from control path implementations. In order to reduce packet and event processing time (not simulation time), the event scheduler and the basic network component objects in the data path are written and compiled using C++. These compiled objects are made available to the OTcl interpreter through an OTcl linkage that creates a matching OTcl object for each of the C++ objects and makes the control functions and the configurable variables specified by the C++ object act as member functions and member variables of the corresponding OTcl object. In this way, the controls of the C++ objects are given to OTcl. It is also possible to add member functions and variables to a C++ linked OTcl object. The objects in C++ that do not need to be controlled in a simulation or internally used by another object do not need to be linked to OTcl. Likewise, an object (not in the data path) can be entirely implemented in OTcl. Figure 2 shows an object hierarchy example in C++ and OTcl. One thing to note in the figure is that for C++ objects that have an OTcl linkage forming a hierarchy, there is a matching OTcl object hierarchy very similar to that of C++.

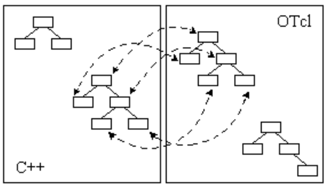


Fig. 2

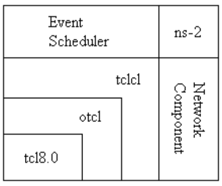
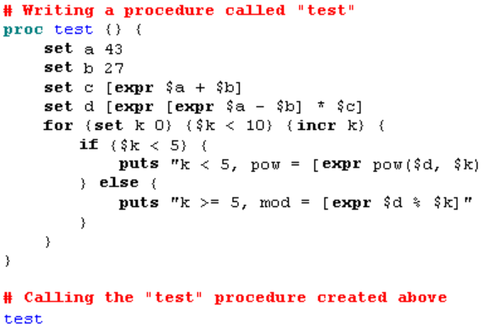


Fig. 3

B. OTcl: The User Language

As mentioned in the overview section, NS is basically an OTcl interpreter with network simulation object libraries. It is very useful to know how to program in OTcl to use NS. This section shows an example Tcl and OTcl script, from which one can get the basic idea of programming in OTcl. This section and the sections after assumes that the reader installed NS, and is familiar with C and C++.

Example 1 is a general Tcl script that shows how to create a procedure and call it, how to assign values to variables, and how to make a loop. Knowing that OTcl is Objectorieneted extension of Tcl, it is obvious that all Tcl commands work on OTcl the relationship between Tcl and Otcl is just same as C and C++. To run this script you should download ex-tcl.tcl, and type "ns ex-tcl.tcl" at your shell prompt. The command "ns" starts the NS (an OTcl interpreter). You will also get the same results if you type "tcl ex-tcl.tcl", if tcl8.0 is installed in your machine.

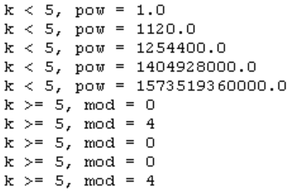


```
# Writing a procedure called "test"
proc test {} {
    set a 43
    set b 27
    set c [expr $a + $b]
    set d [expr [expr $a - $b] * $c]
    for {set k 0} {$k < 10} {incr k} {
        if {$k < 5} {
            puts "k < 5, pow = [expr pow($d, $k)]"
        } else {
            puts "k >= 5, mod = [expr $d % $k]"
        }
    }
}

# Calling the "test" procedure created above
test
```

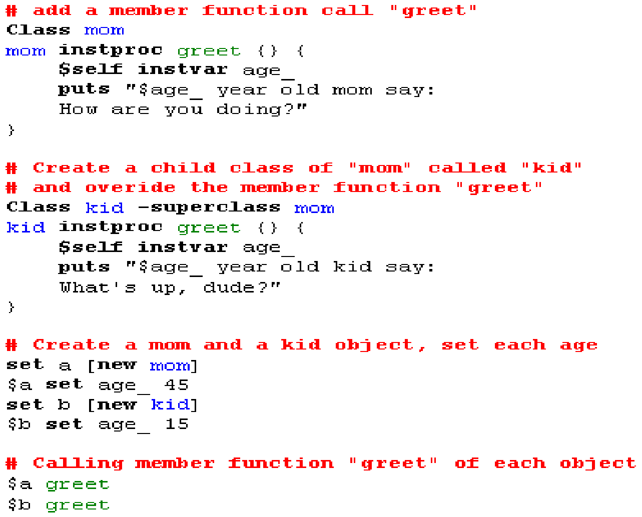
Example 1. A sample Tcl scripts

The keyword puts prints out the following string within double quotation marks. The following shows the result of Example 1



```
k < 5, pow = 1.0
k < 5, pow = 1120.0
k < 5, pow = 1254400.0
k < 5, pow = 1404928000.0
k < 5, pow = 1573519360000.0
k >= 5, mod = 0
k >= 5, mod = 4
k >= 5, mod = 0
k >= 5, mod = 0
k >= 5, mod = 4
```

The next example is an objectoriented programming example in OTcl. This example is very simple, but shows the way which an object is created and used in OTcl. As an ordinary NS user, the chances that you will write your own object might be rare. However, since all of the NS objects that you will use in a NS simulation programming, whether or not they are written in C++ and made available to OTcl via the linkage or written only in OTcl, are essentially OTcl objects, understanding OTcl object is helpful.



```
# add a member function call "greet"
Class mom
mom instproc greet {} {
     $self instvar age_
     puts "$age_ year old mom say:
     How are you doing?"
}

# Create a child class of "mom" called "kid"
# and overide the member function "greet"
Class kid -superclass mom
kid instproc greet {} {
     $self instvar age_
     puts "$age_ year old kid say:
     What's up, dude?"
}

# Create a mom and a kid object, set each age
set a [new mom]
$a set age_ 45
set b [new kid]
$b set age_ 15

# Calling member function "greet" of each object
$a greet
$b greet
```

Example 2. A Sample OTcl Script

Example 2 is an OTcl script that defines two object classes, "mom" and "kid", where "kid" is the child class of "mom", and a member function called "greet" for each class. After the class definitions, each object instance is declared, the "age" variable of each instance is set to 45 (for mom) and 15 (for kid), and the "greet" member function of each object instance is called. The keyword Class is to create an object class is to define a member function to an object class. Class inheritance is specified using the keyword –super class. In defining member functions, $self acts same as the "this" pointer in C++, checks if the following variable name is already declared in its class or in its super class. If the variable name given is already declared, the variable is referenced, if not a new one is declared. Finally, to create an object instance, the keyword new is used as shown in the example. Downloading ex-otcl.tcl and executing "ns ex-otcl.tcl" will give you the following result:

45 year old mom say:

How are you doing?

15 year old kid says:

What's up, dude?

C. Event Scheduler

This section talks about the discrete event schedulers of NS. As described in the Overview section, the main users of an event scheduler are network components that simulate packet handling delay or that need timers. Figure 4 shows each network object using an event scheduler. Note that a network object that issues an event is the one who handles the event later at scheduled time. Also note that the data path between network objects is different from the event path. Actually, packets are handed from one network object to another using send (Packet* p) {target_-> receiver (p)}; method of the sender and receiver (Packet*, Handler* h = 0) method of the receiver.

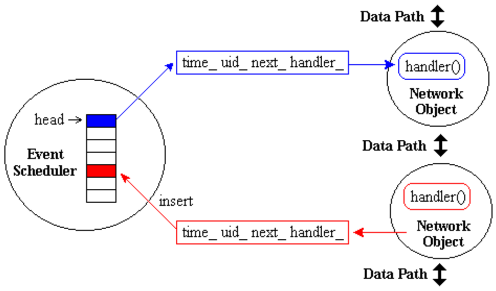


Fig. 4 Discrete event scheduler

NS has two different types of event schedulers implemented. These are real-time and non-real-time schedulers. For a non-real-time scheduler, three implementations (List, Heap and Calendar) are available; even though they are all logically perform the same. This is because of backward compatibility: some early implementation of network components added by a user (not the original ones included in a package) may use a specific type of scheduler not through public functions but hacking around the internals. The Calendar non-real- time scheduler is set as the default. The real time scheduler is for emulation, which allows the simulator to interact with a real network. Currently, emulation is under development although an experimental version is available. The following is an example of selecting a specific event scheduler.

Set ns [new simulator]
$ns   Use-scheduler Heap

IV. System Design

A. Modules

- Network Formation
- Path Finding and data sending using RLNC
- Wormhole Detecting

B. Network Formation

In region each node sends —hello message to other nodes which allows detecting it. Once a node detects —hello message from another node (neighbor), it maintains a contact record to store information about the neighbor. Using multicast socket, all nodes are used to detect the neighbor nodes. Once after finding neighbor nodes a queue is maintained for each neighboring node called as real queue.

C. Path Finding and Data Sending Using RLNC

In RLNC technique we sending packet in multiple path in order to best utilize resources, before data transmissions, routing decisions (i.e., how many times of transmissions a forwarder should make for each novel packet) are made based on local link conditions by some test transmissions. We find redundant path, it means which is an unwanted path.

D. Wormhole Deecting and Remove The Link

We detect wormhole link in wireless network coding systems, where no fixed routes exist, ETX, the expected number of the packets for the source node to transmit so that the target node (intermediate node or recipient) receives the packet, provides a way to portray the topological structure of the network and the relations among nodes. After detecting the Wormhole link it will be removed in the network.

V. Network Components

This section talks about the NS components, mostly compound network components, a partial OTcl class hierarchy of NS, which will help understanding the basic network components.

The root of the hierarchy is the Tcl Object class that is the super class of all OTcl library objects (scheduler, network components, timers and the other objects including NAM related ones). As an ancestor class of Tcl Object, Ns object class is the super class of all basic network component objects that handle packets, which may compose compound network objects such as nodes and links. The basic network components are further divided into two subclasses, Connector and Classifier, based on the number of the possible output data paths. The basic network objects that have only one output data path are under the Connector class, and switching objects that have possible multiple output data paths are under the Classifier class.

Node and Routing

A node is a compound object composed of a node entry object and classifiers there are two types of nodes in NS. A unicast node has an address classifier that does unicast routing and a port classifier. A multicast node, in addition, has a classifier that classify multicast packets from unicast packets and a multicast classifier that performs multicast routing.

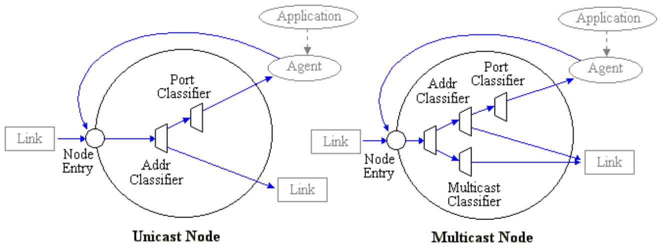


Fig. 5. Node and routing.

In NS, Unicast nodes are the default nodes. To create Multicast nodes the user must explicitly notify in the input OTcl script, right after creating a scheduler object, that all the nodes that will be created are multicast nodes. After specifying the node type, the user can also select a specific routing protocol other than using a default one.

Unicast

-$ns rtproto type

- type: Static, Session, DV, cost, multi-path

Multicast

-$ns multicast (right after set $ns [new Scheduler]) $ns mrtproto type

- type: CtrMcast, DM, ST, BST

A. Packet Flow Example

Until now, the two most important network components (node and link).The network consist of two nodes (n0 and n1) of which the network addresses are 0 and 1 respectively. A TCP agent attached to n0 using port 0 communicates with a TCP sink object attached to n1 port 0. Finally, an FTP application (or traffic source) is attached to the TCP agent, asking to send some amount of data.

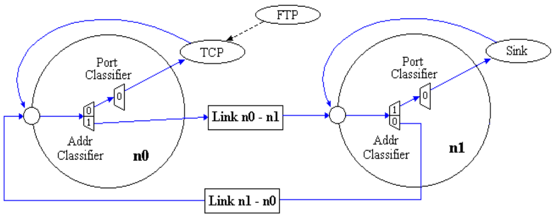


Fig. 6. Packet flow

VI. Overall Description

A. Purpose

The main aim of this project is how to detect a wormhole attack using Expected Transmission count technique in Random linear Network coding (RLNC) System .Wormhole link will be detected in the network and packets are transmitted through secure link**.**

B. Project Scope

To improve the system performance of wireless Network, network coding is shown to be effective approach and it is totally different from traditional network. In traditional network where intermediate nodes store and forward packets as the original. In contrast, in wireless network coding systems, the forwarders are allowed to apply encoding schemes on what they receive, and thus they create and transmit new packets.

In a wormhole attack, the attacker can forward each packet using wormhole links and without modifies the packet transmission by routing it to an unauthorized remote node. Hence, receiving the rebroadcast packets by the attackers, some nodes will have the illusion that they are close to the attacker. With the ability of changing network topologies and bypassing packets for further manipulation, wormhole attackers pose a severe threat to many functions in the network, such as routing an localization. If wormhole attacks are launched in routing, the nodes close to attackers will receive more packets than they should and be considered as having a good capability in help forwarding packets. Thus they will be assigned with more responsibility in packet forwarding than what they can actually provide.

C. External Interface Requirements

a) User interfaces

a) Graphical User Interfaces not in this product.
b) Users are communicated with Buttons with network animator.

b) Hardware interfaces

Linux environment of system and basic need of system feature like random access memory etc

c) Software interfaces

- This software is interacted with the TCP/IP protocol.
- This product is interacted with the and linux
- This product is interacted with the Server Socket
- This product is interacted with TCL

D. Communication Interfaces

The TCP/IP protocol will be used to facilitate communications between the nodes.

E. Performance Requirements

The performance of the wireless sensor network, to execute this project on LAN or wifi communication channel. So we need to one or more than machine to execute the demo. Machine needs the enough hard disk space to install the software and run our project.

Security requirements

- Do not block the some available component through the Linux firewall
- Do not block the some available component in network-simulator 2

F. Software Qulity Attributes

- *Functionality*: are the required functions available, including Interoperability and security
- *Reliability*: maturity, fault tolerance and recoverability.
- *Usability*: how easy it is to understand, learn and operate the software system.
- *Efficiency*: performance and resource behavior.
- *Maintainability*: how easy is it to modify the software
- *Portability*: can the software easily be transferred to another environment, including install ability.

G. Product Features

In this project we detect a wormhole attack. Packet will be sending using RLNC technique, when the attacker attack the nodes while sending data. We find Critical Nodes and then we remove that path. Packet will be sending other paths and expected transmission count is to be increased if wormhole attack is detected.

We first propose a centralized algorithm to detect wormholes leveraging a central node in the network. For the distributed scenarios, we propose a distributed algorithm, DAWN, to detect wormhole attacks in wireless intra-flow network coding systems. The main idea of our solutions is that we examine the order of the nodes to receive the innovative packets in the network. Innovative Packet means intermediate node receives a packet which is linearly independent from previous packets.

H. User Classes and Characteristics

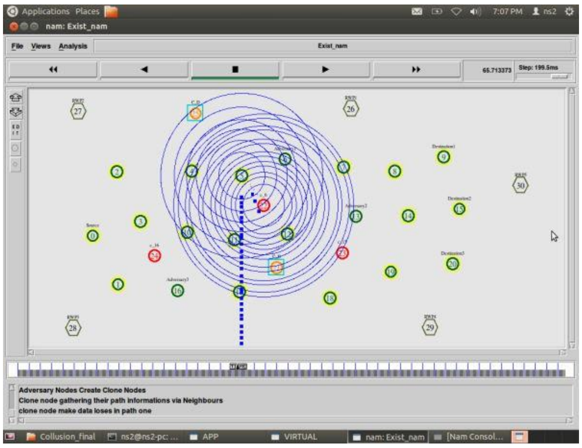
The user privileges vary according to their designations. Basic knowledge of using computers is adequate to use this application. Knowledge of how to handle the system is necessary. The user interface will be friendly enough to guide the user.

I. System Features

In this project we detect a wormhole attack. Packet will be sending using RLNC technique, when the attacker attack the nodes while sending data. We find Critical Nodes and then we remove that path. Packet will be sending other paths and expected transmission

VII. Simulated Output

A. Existing Model

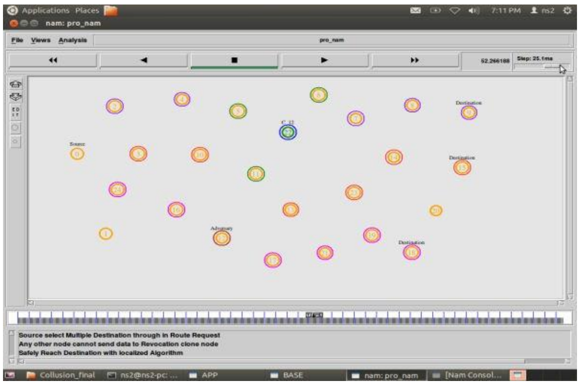


Adversary Nodes Create Clone Nodes
Clone node gathering their path informations via Neighbours
clone node make data loses in path one

Fig. 7. Existing model

- The above diagram represents the existing model.In this theory there exists packet losses in transmission.
- They are overcome by the proposed model.

B. Proposed Model

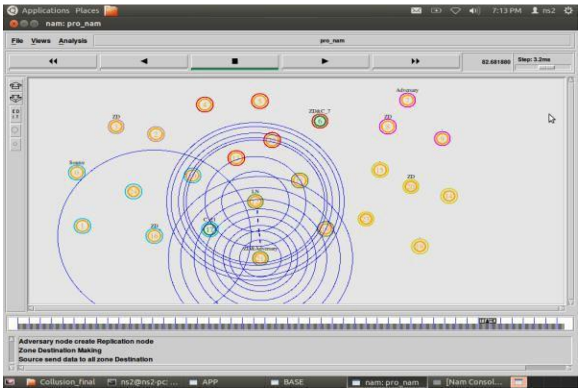


Source select Multiple Destination through in Route Request
Any other node cannot send data to Revocation clone node
Safely Reach Destination with localized Algorithm

Fig. 8. Proposed models

- This overcomes the drawbacks of the basic model
- In this system we introduce single source and single destination method by which we are able to receive and transmit packets without any loss.
- Only drawback is that we are unable to use the other path for communication, thereby the path remains idle.

C. *Enhanced Model*



Adversary node create Replication node
Zone Destination Making
Source send data to all zone Destination

Fig. 9. Enhanced models

- In this model we are able to access the two or more paths to transmit the packet(single source and multiple destination)
- Therefore all paths are used effectively in this model.

D. *Delay Ratio*

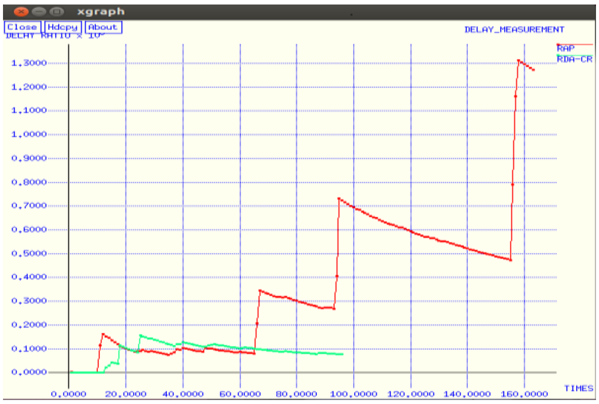


Fig. 10. The delay is analyzed by considering time and delay ratio

E. *Total Loss*

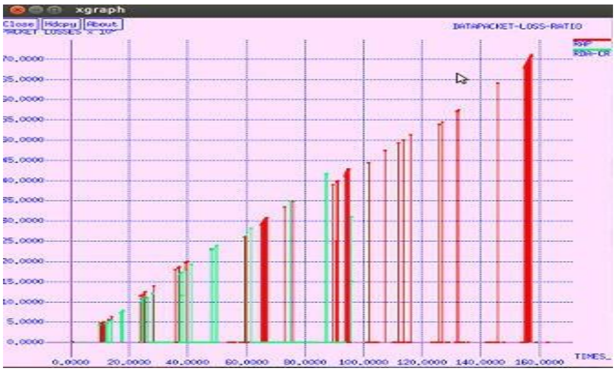


Fig. 11. The total loss is analyzed by considering time

F. *Throughput Workload*

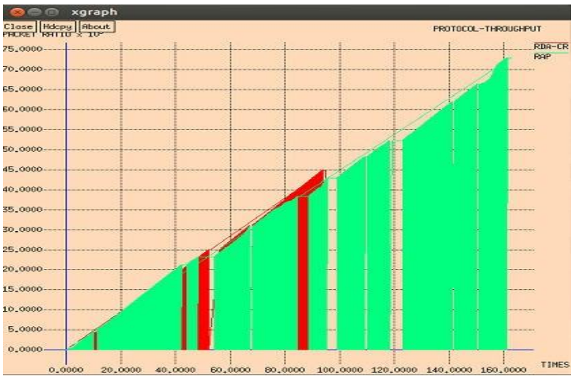


Fig. 12. The throughput workload is analyzed by considering time and throughput.

G. *Protocol Frequency*

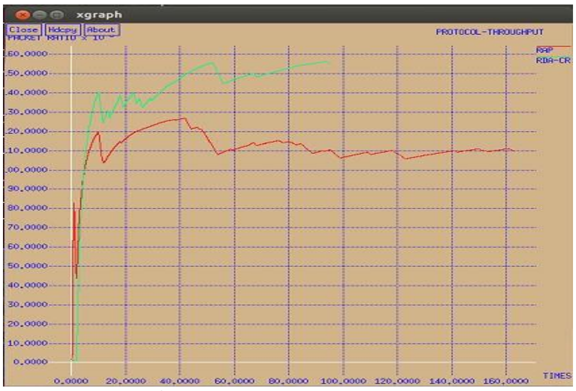


Fig. 13. The protocol frequency is analyzed by considering time and phase

H. *Source Frequency*

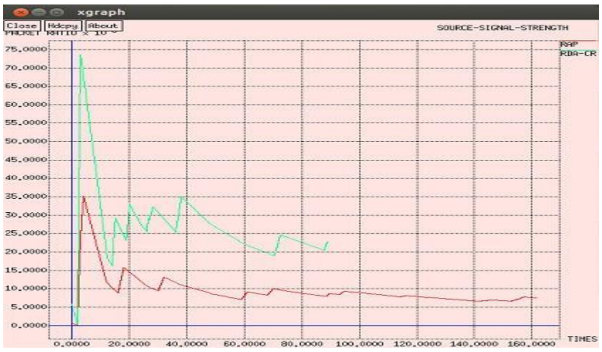


Fig. 14. The source frequency is analyzed by considering time and signal strength

VIII. CONCLUSION

We have proposed a Centralized Algorithm that assigns a central node to collect and analyze the forwarding behaviors of each node in the net- work, in order to react timely when wormhole attack is initiated. We have proven the correctness of the Centralized Algorithm by deriving a lower bound of the deviation in the algorithm. We have also proposed a Distributed detection Algorithm against Wormhole in wireless Network coding systems.

REFERENCES

[1] D. Dong, Y. Liu, X. Li, and X. Liao, "Topological detection on wormholes in wireless ad hoc and sensor networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 6, pp. 1787–1796, 2011.

[2] J. Kim, D. Sterne, R. Hardy, R. K. Thomas, and L. Tong, "Timingbased localization of in-band wormhole tunnels in MANETs," in *Proceedings of the Third ACM Conference on Wireless Network Security, WISEC*, pp. 1–12, 2010.

[3] R. Poovendran and L. Lazos, "A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks," *Wireless Networks*, vol. 13, no. 1, pp. 27–59, 2007.

[4] S. Biswas and R. Morris, "Opportunistic routing in multihop wireless networks," *ACM SIGCOMM Computer Communication Review*, vol. 34, issue 1, pp. 69–74, 2004.

[5] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 169–180, 2007.

[6] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcrof," XORs in the air: Practical wireless network coding," in *Proceedings of the 2006 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 243–254, 2006.

[7] S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[8] S. R. D. R. Maheshwari, and J. Gao, "Detecting wormhole attacks in wireless networks using connectivity information," *26th IEEE International Conference on Computer Communications*, pp. 107–115, 2007.

[9] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[10] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole attacks in wireless networks," *IEEE Journal on Selected Areas in Communications*., vol. 24, no. 2, pp. 370–380, 2006.