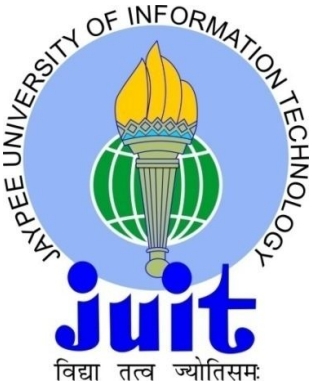
DETECTION OF SYBIL ATTACK IN A MOBILE AD HOC NETWORK (MANET)

Ratish Goel	**071411**
Ankur Sharma	**071414**
Tashi Weezer	**071424**

Under the Supervision of

Mr. Amol Vasudeva



May 2011

Submitted in partial fulfillment of the Degree of
BACHELOR OF TECHNOLOGY
(CSE & IT)

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT,

SOLAN – 173234, HIMACHAL PRADESH,

INDIA

TABLE OF CONTENTS

Topics	Page No.
CERTIFICATE	**III**
ACKNOWLEDGMENT	**IV**
ABSTRACT	**V**
LIST OF ABBREVIATIONS	**VI**
LIST OF FIGURES	**VII**
Chapter 1 Introduction	**1**
1.1 Introduction	1
1.2 Wireless Mobile Ad-hoc Networks (MANET)	2
1.3 Characteristics of Ad hoc networks	4
1.4 Applications of MANET	4
1.5 Design issues and constraints in Ad-hoc networks	6
Chapter 2 Security Issues	**9**
2.1 Network Security	9
2.2 Type of attacks in an Ad hoc network	9
2.2.1 Impersonation/ spoofing attacks	9
2.2.2 Sinkhole attacks	9
2.2.3 Wormholes	10
2.2.4 Sleep deprivation torture	10
2.2.5 Rushing attack	10
2.2.6 Denial of Service (DoS) and Flooding	10
2.2.7 Sybil attack	10
Chapter 3 Attacks	**11**
3.1 Sybil Attack in MANET	11
3.2 Different Attack Scenarios	11
3.2.1 Basic Scenario: no Sybil attack	12
3.2.2 Scenario 1: Localized Sybil attack	12
3.2.3 Scenario 2: Public Sybil attack	12
3.2.4 Scenario 3: Localized Sybil attack by disabling nodes	13
3.2.5 Scenario 2: Public Sybil attack by disabling nodes	14
3.3 Sybil Attack Detection Models	14
3.4 Approaches to detect some attacks	15
3.4.1 Watchdog Mechanism	15
3.4.2 Improved Watchdog-like Mechanism	15
3.4.3 Anomaly detection features	15

3.4.4 Collaboration of misuse Watchdog in neighbor 16
3.4.5 Extended Watchdog misuse feature 16
3.4.6 Neighboring Nodes confirmation 16

Chapter 4 Detection Mechanism **17**
4.1 Detection of Sybil attack in a MANET using neighboring information 17
4.2 Assumption and attack model 17
4.2.1 Detecting of Sybil nodes 18
4.2.2 Corresponding notation 18
4.2.3 Determine each Common Neighbor 19
4.2.4 Procedure for detecting Sybil nodes in MANET 20
4.3 Algorithm for detecting Sybil nodes in MANET 21
4.4 Flowchart 23
4.5 Enhancement Mechanism 24

Chapter 5 Implementation **25**
5.1 Network Simulator (ns2) 25
5.1.1 Architecture of NS2 26
5.1.2 Visualization the network flow 30
5.1.3 Analyzing: Trace file format 30
5.2 Programming 31
5.2.1 Ns2 code 31
5.2.2 Screenshot NAM 36
5.2.3 C++ code 37
5.2.4 Screenshot C++ result 47

Conclusion **48**

Appendix A Installing NS2 in LINUX platform **49**

Bibliography **51**

CERTIFICATE

This is to certify that project report entitled "**Detection of Sybil attack in a Mobile Ad Hoc Network (MANET)**", submitted by **Ratish Goel, Ankur Sharma** and **Tashi Weezer** in partial fulfillment for the award of degree of Bachelor of Technology in Information Technology of Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor ……..

Name of Supervisor: Mr. Amol Vasudeva

Designation: Lecturer, Department of CSE & IT, JUIT

Date:

ACKNOWLEDGEMENT

The project entitled as Detection of Sybil attack in Mobile ad hoc network using neighboring information is used to detect malicious intrusion in a network, henceforth limiting the security risks. This project is the result of our team cooperation and endurance in best of ways for these couple of months.

Our project guide Mr. Amol Vasudeva who is an esteemed Lecturer for Department of Computer Science and IT, JUIT, has indeed guided us to proceed this project in best of his capability and enlightening us with his knowledge and wisdom. He has given us the push that we need to go further and do more. So we take this opportunity to express our sincere gratitude and thanks to him for his lending help and cooperation.

Signature of the student:	……..	……..	……
Name of Student:	Ratish Goel	Ankur Sharma	Tashi Weezer

Date:

SUMMARY

Our key issue will be to detection of Sybil attack in a MANET (Mobile Ad hoc Network). Out of many detection methods and techniques that exist, we will emphasis on detection by neighboring information.

It states that no two neighbors in a highly populated area will have same set of neighbors in their vicinity and this is true in even real life scenario.

Identifying the set of common neighbors will help us detect which are Sybil nodes and malicious node and thus be sure that rest of the nodes are normal nodes through which we can safely send our message to destination of our choice without any security risks like eavesdropping or manipulation of vital information in the message.

_____
Signature of Student
Name: Ratish Goel
Date:

_____
Signature of Student
Name: Ankur Sharma
Date:

_____
Signature of Student
Name: Tashi Weezer
Date:

_____
Signature of Supervisor
Name: Mr. Amol Vasudeva
Date:

LIST OF ABBREVIATIONS

TCP/IP - Transmission Control Protocol. It provides a reliable, ordered delivery of stream of bytes from a program on one computer to another program on another computer

UDP - User Datagram Protocol. It provides an unreliable service and datagrams may arrive out of order, appear duplicated or go missing without notice

FTP - File Transfer Protocol. It is a standard network protocol used to exchange and manipulate files over an Internet Protocol computer network, such as Internet

CBR – Constant Bit Rate

DSR – Dynamic Source Routing

DSDV – Destination Sequenced Distance Vector

ADOV – Ad-hoc On Demand Distance Vector

QoS – Quality of Service

DoS - Denial of Service

LIST OF FIGURES

Figures **Page No.**

Figure 1: Mobile Ad-hoc Network 3
Figure 2: Simple Wireless 11
Figure 3: Localized Sybil attack by nodes 12
Figure 4: Public Sybil attack by nodes creation 12
Figure 5: Abstract representation of the Sybil node 13
Figure 6: Counterfeiting identities 1 13
Figure 7: Counterfeiting identities 2 14
Figure 8: Victim node collecting neighboring node data 19
Figure 9: Illustration of Detection method 22
Figure 10: Flow chart for detecting Sybil nodes 23
Figure 11: Enhance detection mechanism 24
Figure 12: ns Directory structure 25
Figure 13: Basic architecture of NS 26
Figure 14: OTcl and C++: The Duality 26
Figure 15: Nam Console 30
Figure 16: Trace file structure 30
Figure 17: Wireless network 36
Figure 18: Wireless network: Computing nearest neighbor 36
Figure 19: Packet flow 37
Figure 20: Nodes generated by random function 47
Figure 21: Detection of Sybil nodes 47

CHAPTER 1
INTRODUCTION

1.1 Introduction

Wireless communication between mobile users is becoming more than ever before. This leap is due to recent technological advances in laptop computers and wireless data communication devices, such as wireless modems and wireless LAN's and many more. Now there are technologies with lower price and higher data rate than before and that why mobile computing is continues to growing bigger and sophisticated.

There are two distinct approaches for transmitting in a wireless communication between two hosts.

Two types of approach for wireless communication:

1. *Cellular Network infrastructure*
2. *Ad-hoc network*

The first approach is to let the existing cellular network infrastructure carry data as well as voice. The major problems are there should not be any delay or packet loss when connection is switching from one connection to another. In such approach it is limited to some space that is network can exist only where cellular network infrastructure stands.

The second approach is to form an ad-hoc network among all the users wanting to communicate with each other. This means that all users participating in the ad-hoc network must be willing to forward data packets and make sure that the packets are delivered from source to destination but this form of networking is limited in range by the individual nodes transmission ranges and is typically smaller compared to the range of cellular systems. This does not mean that the cellular approach is better than the ad-hoc approach.

1.2 Wireless Mobile Ad Hoc Network (MANET)

A wireless ad-hoc network is a collection of mobile/semi-mobile nodes with no pre-established infrastructure forming a temporary network. Each of the nodes has a wireless interface and communicates with each other over either radio or infrared.

In general, mobile ad hoc networks are formed dynamically by an autonomous system of mobile nodes that are connected via wireless links without using an existing network infrastructure or centralized administration. The nodes are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Mobile ad hoc networks are infrastructure less networks since they do not require any fixed infrastructure such as a base station for their operation. In general, routes between nodes in an ad hoc network may include multiple hops and, hence, it is appropriate to call such networks "multi-hop wireless ad hoc networks".

Examples of node in an ad-hoc network are laptop computers and personal digital assistants (mobile phones, handheld devices) that communicate directly with each other.

Advantage of Ad-hoc network:

- On demand setup
- Fault tolerance
- Unconstrained connectivity.

Figure below show the ad hoc network where each node will be able to communicate directly with other nodes that reside within its transmission range. For communicating with nodes that reside beyond this range, the node needs to use intermediate nodes to rely messages hop by hop.

For example if node A want to communicate with node F which is on the outermost ends, that is it is out of communication range of each other. So it will use the middle node B and node D which will act as routers to forward the message from source node A to destination node F. The middle node and the three nodes have formed an ad-hoc network.

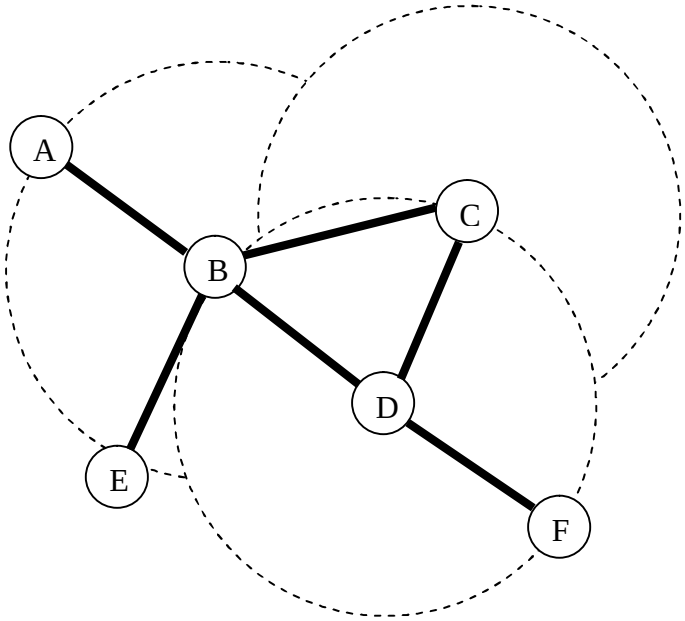


Figure 1: Mobile Ad –hoc Network

An ad-hoc network uses no centralized administration and this is done to make sure that the network won't collapse just because one of the mobile nodes moves out of transmitter range of the other. Nodes should be able to enter/leave the network as they wish. Due to the limited transmitter range of every node, multiple hops may be needed to reach other nodes which are at the other side and every node wishing to participate in an ad-hoc network must be willing to forward packets for other nodes such that no data loss is there. Thus every node should acts as a host and as well as a router.
A router is an entity, which, among other things runs a routing protocol and a mobile host is simply an IP-addressable host/entity.

Ad-hoc networks are also capable of handling topology changes and malfunctions in nodes. It is fixed through network reconfiguration. For instance, if a node leaves the network and causes link breakages between any nodes then the affected nodes can easily request new routes and the problem will be solved. But this will slightly increase the delay time, but the network will still be operational.

Wireless ad-hoc networks take advantage of the nature of the wireless communication medium. So it is not constrained like in a wired network the physical as every connection will need permission for communication. This restriction is not present in the wireless domain and, provided that two nodes are within transmitter range of each other, an instantaneous link between them may form.

1.3 Characteristics of Ad Hoc Networks

- *Wireless*: Nodes communicate wirelessly that is without any centralized infrastructure and share the same media for communication (radio, infrared, etc.).

- *Ad hoc based*: A mobile ad hoc network is a temporary network, formed by collection of nodes which are mobile in nature.

- *Autonomous and infrastructure less*: MANET does not depend on any established infrastructure or centralized administration. Each node operates in distributed peer-to-peer mode, acts as an independent router, and generates independent data.

- *Multihop routing*: No dedicated routers are necessary; every node acts as a router and forwards each other's packet to enable information sharing between mobile hosts.

- *Mobility*: Each node is free to move about while communicating with other nodes. The topology of such an ad hoc network is dynamic in nature due to constant movement of the participating nodes, causing the intercommunication patterns among nodes to change continuously.

1.4 Application of MANET

Ad hoc networks are flexible networks that can be setup anywhere at any time, without any infrastructure including pre-configuration or administration. That why it have great commercial potential and advantages that mobile ad hoc networking have.

Following range of mobile ad hoc network application:

Emergency services

- Search and recuse operation as well as disaster recovery. e.g., early retrieval and transmission of patient data (record, status, diagnosis) from/to the hospital.
- Replacement of a fixed infrastructure in case of earthquakes, hurricanes, fire, etc.

Commercial environments

- *E-Commerce,* e.g., electronic payments from anywhere and any place.
- *Business:*

  Dynamic access to customer files stored in a central location on the fly provide consistent databases for all agents mobile office
- *Vehicular Services:*

  Transmission of news, road conditions, weather, music
  Local ad-hoc network with nearby vehicles for road/accident guidance.

Home and enterprise networking

- Home/office wireless networking (WLAN) e.g., shared wide range of applications; use PDA to print from anywhere, etc.
- Personal area network (PAN).

Educational applications

- Set up virtual classrooms or conference rooms.
- Set up ad hoc communication during conferences, meetings, or lectures

Entertainment

- Multi-user games
- Robotic pets
- Outdoor Internet access

Locations-aware services

- Follow-on services, e.g., automatic call forwarding, transmission of the actual workspace to the current location.
- *Information services*:

  Push, e.g., advertise location-specific service, like gas stations
  Pull, e.g., location-dependent travel guide; services (printer, fax, phone, server, gas stations) availability information; caches, intermediate results, state information, etc.

Tactical network

- Military communication, operations
- Automated Battlefields

1.5 Design Issues and Constraints

That ad hoc architecture has many benefits such as self-configuration, ease of deployment and so on but it also has some drawback in a network. Ad hoc wireless networks inherit the traditional problems of wireless communications, such as bandwidth optimization, power control, and transmission quality enhancement, while, in addition, their mobility, multi-hop nature, and the lack of fixed infrastructure create a number of complexities and design constraints that are new to mobile ad hoc networks.

Following are limitation of Ad hoc network:

- **Infrastructure less:**

Ad hoc network lack fixed infrastructure and in addition there are design issues related with wireless connection compared to a fixed network. Difficulty in handling fault detection and management as no centralized entity exist (Distributed network management across different nodes).

- **Dynamic network topologies:**

As nodes are dynamic in nature, the network topology always changes with time, leading to many re-routing routes and network changes may cause packet losses.

- **Physical layer limitation:**

The radio interface at each node uses broadcasting for transmitting traffic and usually has limited wireless transmission range, resulting in specific mobile ad hoc network problems like hidden terminal problems, exposed terminal problem, and so on. Collisions are inherent to the medium, and there is a higher probability of packet losses due to transmission errors compared to wire line systems.

- **Bandwidth constrain:**

As mobile nodes communicate with each other via wireless, issues such as limited bandwidth, variable capacity of data transfer, error prone and insecure wireless channel persist. Lower channel capacity causes congestion in the network.

- **Variation in like and Node capabilities:**

Designing network protocols and algorithms for heterogeneous network can be complex, requiring dynamic adaption to changing power and channel conditions, congestion and service environment.

- **Energy constrain:**

Each mobile node has limited power capacity and processing power is less and can in turn cause limitation for services and application. Also each node acts as both end system as well as a router.

- **Network Robustness and Reliability:**

In MANET, network connectivity is obtained by routing and forwarding among nodes. Although this replaces the constraints of fixed infrastructure connectivity, it also brings design challenges. Due to various conditions like overload, acting selfishly, or having broken links, a node may fail to forward the packet. Misbehaving nodes and unreliable links can have a severe impact on overall network performance. Lack of centralized monitoring and management points means these types of misbehaviors cannot be detected and isolated quickly and easily, adding significant complexity to protocol design.

- **Network scalability:**

Current popular network management algorithms were mostly designed to work on fixed or relatively small wireless networks. Many mobile ad hoc network applications involve large networks with tens of thousands of nodes, as found, for example, in sensor networks and tactical networks. Scalability is critical to the successful deployment of such networks. The evolution toward a large network consisting of nodes with limited resources is not straightforward and presents many challenges that are still to be solved in areas such as addressing, routing, location management, configuration management, interoperability, security, high-capacity wireless technologies, and so on.

- **Quality of Service (QoS)**

A quality of service (QoS) guarantee is essential for successful delivery of multimedia network traffic. QoS requirements typically refer to a wide set of metrics including throughput, packet loss, delay, jitter, error rate, and so on . Wireless and mobile ad hoc specific network characteristics and constraints described above, such as dynamically changing network topologies, limited link bandwidth and quality, variation in link and node capabilities, pose extra difficulty in achieving the required QoS guarantee in a mobile ad hoc network.

- **Network security:**

Mobile wireless networks are vulnerable to information and physical security threats and attacks as network works on opened and shared broadcast wireless channels. MANET being a distributed infrastructure less network means that centralized security control is hard to implement and must rely on individual security solutions.

Furthermore Ad hoc network are designed for specific environment and must operate with full availability even in adverse condition so security solution applied like in traditional network may not be suitable.

Some key security requirements in ad hoc networking include:

- *Confidentiality:* preventing passive eavesdropping.
- *Access control:* protecting access to wireless network infrastructure.
- *Data integrity:* preventing tampering with traffic (i.e., accessing, modifying or injecting traffic).

Biggest advantage as well as disadvantage is anyone with proper hardware and knowledge about the network topology and protocol can connect to the network.

CHAPTER 2
SECURITY ISSUES

2.1 Network Security

Ad hoc network has to rely on individual security solution unlike infrastructure networking. Understanding the possible forms of attacks is the first step for developing security solution. Nodes being dynamic cause a serious problem as it might disappear anytime so this makes easier for attackers to disturb the network.

Classifications of attacks
External attacks are committed by parties which are not a part of the network.
Internal attacks are sourced from inside a network and a part of the network.

Passive attacks don't take part on attacking any node or disturbing the service by just eavesdrops on the communication in context for stealing information.
Active attacks try to disturb follow of service and alter data.

Example: Student network using PDAs which are interconnected and battlefield scenario where soldier are connected by wireless communication device. (Defining security measures needed?).

2.2 Attack Types:

2.2.1 Impersonation/ spoofing attacks

Faking an identity of a node thus stealing vital information. Also called malicious node might obstruct flow by injecting fault routing packets into the network or by modifying routing information.
Measures: Strong data encryption, good authentication algorithms and secure routing protocols.

2.2.2 Sinkhole attacks

Attract data to itself from neighboring node.
Giving access to the data to the node it is basic for many attacks like eavesdropping and data alteration.Present themselves as most suitable partner for multi hop routing. (This creates a loophole in routing algorithms).

Measures: Using multi-path sends data redundantly rather than relying on one path only and probabilistic routing protocols measure the probability of packet reaching the other end.

2.2.3 Wormholes

Fake the route of the packet takes thus taking less time to reach its destination. This is dangerous as it confuse the routing mechanism which relies on distance between nodes.
Harmful if data within the packet is altered to contain different information than the original one and relaying it over faster by wormhole attackers. Action taken up is using timestamps and location stamp that is time it would take between sender and receiver and also the geographic location. Each packet will be tagged by it and when it reaches the destination it will be compared with its own time and location stamps.

2.2.4 Sleep deprivation torture

Request a node for a service again and again and not allowing it to go into idle mode or power preserving state thus depriving if from its sleep. Worst case scenario for limited resources like battery power.
Minimizing its effect by setting priority.

2.2.5 Rushing attack

Directed against on-demand routing protocols using Dynamic Source Routing protocol (DSR).
Detected by evaluating of Route Discovery.

2.2.6 Denial of Service(DoS) and Flooding

Malicious node can cause traffic subversion and DoS by modifying routing information and can lead to communication delays.

2.2.7 Sybil Attack

Malicious nodes in a network will impersonate the identity of several nodes and so by doing this the redundancy of many routing protocol is undermined. This is Sybil attack. Multi-hop and other verification are not applicable in case of Sybil attacks.
Some light on public key management (PKM) where third party acts as the one to verify and give access to other nodes.PKM works on based of infrastructure, so prove incapable in MANET. Multi-hop also gives a loop hole here.

CHAPTER 3

ATTACKS

3.1 Sybil Attack

The threat of Sybil attacks was first identified by JR Douceur in the context of peer-to-peer system. Douceur claimed that within such distributed computing environment, a single device could easily project multiple identities due to lack of trusted, central authorities within the network. It was named after subject of a book Sybil, a case study of a woman with multiple personality disorder. Because each entity is only aware of others through messages over a communication channel, a Sybil attacker can assume many different identities by sending messages with different identities. If a single faulty entity can present multiple identities, it can control a substantial fraction of system.

3.2 Different Attack Scenarios

This section presents the scenarios that explain how a node can obtain the multiple identities to conduct the Sybil attack. Each scenario deals with a different form of Sybil attack.

3.2.1 Basic Scenario: no Sybil attack

.

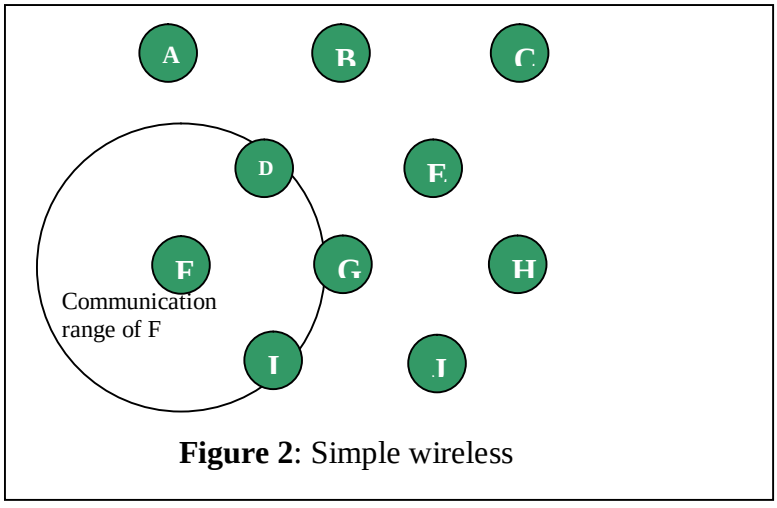


Figure 2: Simple wireless

Neutral ad hoc network: 10 nodes are communicating using the same emission power and antenna and their communication range will be limited by a circle which the rayon will be the distance with the closest neighbors. They are equally distributed on the map as in Figure 2.

3.2.2 Scenario 1: Localized Sybil attack by creating new identities

If one of the nodes is performing a Sybil attack, then this node counterfeits two identities as shown in the Figure 3.

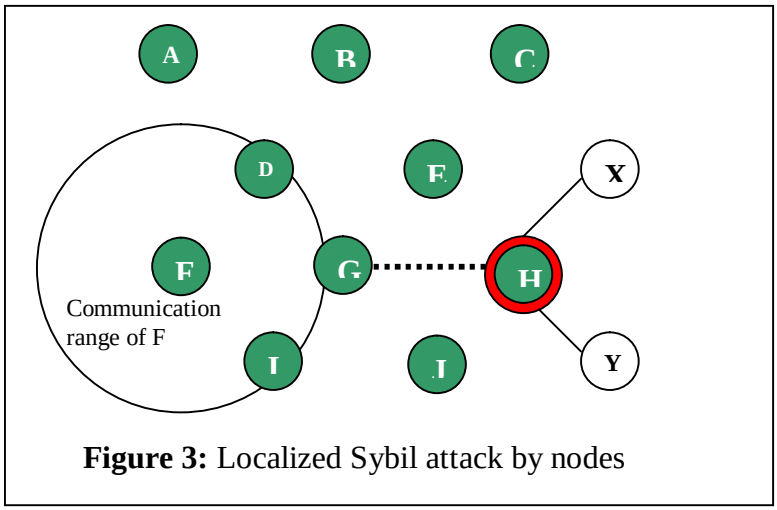


Figure 3: Localized Sybil attack by nodes

This scenario keeps the same previous parameters. In this case, H is performing a Sybil attack on node G by creating two new identities which are X and Y. Nevertheless, only the node G is the victim of the Sybil attack by node H. H does not claim to its other neighbors, E and J, that he knows X and Y. This Sybil attack is called the localized because G claims the identities of E and J only to the node F.

3.2.3 Scenario 2: Public Sybil Attack by creating new identities:

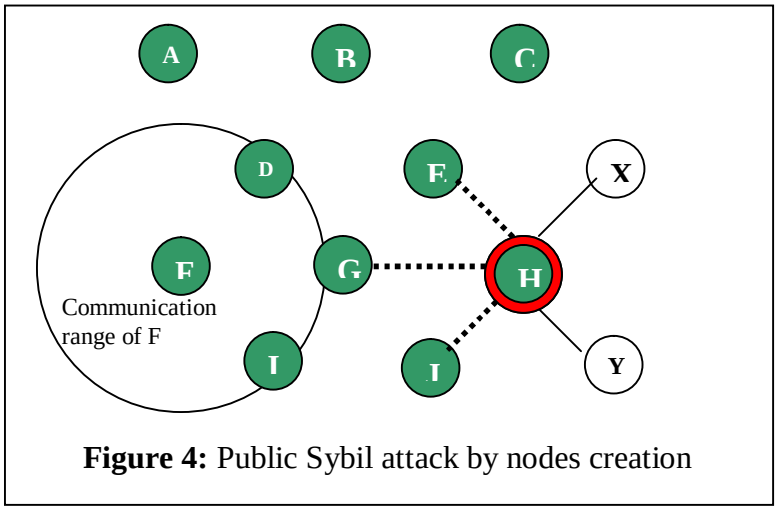


Figure 4: Public Sybil attack by nodes creation

Here it claims to all its neighbors that knows X and Y. The created identities X and Y should have different neighbors than H. For example, in above Figure 4, X should be the neighbor of C. If this is the case, there should be a communication between C and X. However, there won't be any traffic between these two nodes. Indeed, the fake nodes are virtually at the same

place than the compromised node, the attacker representation of the network would look like the following scheme (Figure 5).

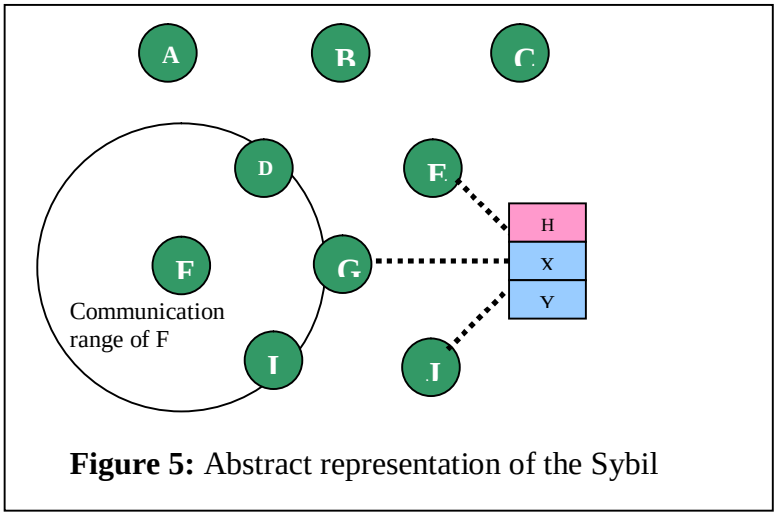


Figure 5: Abstract representation of the Sybil

This could be the interesting feature to detect if a node performs a Sybil attack or not.

3.2.4 Scenario 3: Localized Sybil attack by disabling nodes

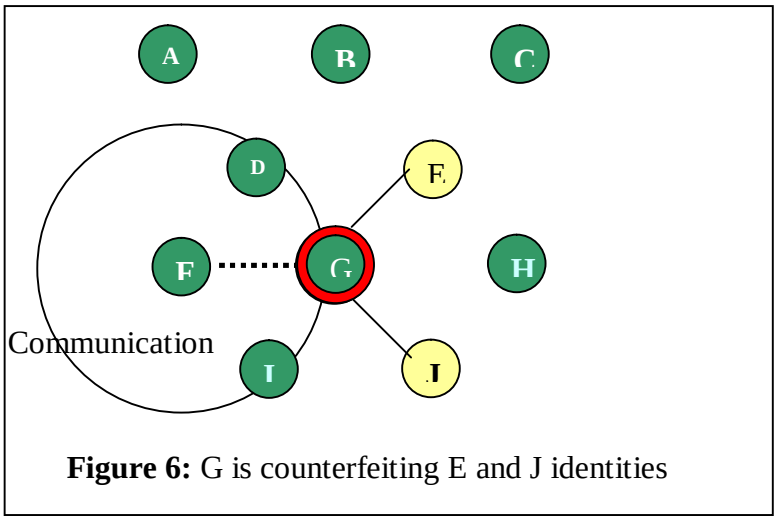


Figure 6: G is counterfeiting E and J identities

In this different scheme, we suppose that G has taken the identities of E and J. The difference with the previous schemes is that E and J were already known by the network. Therefore, in order to perform this type of attack, G will have to disable the real nodes first and then it can counterfeit their identities. This type of Sybil attack is more aggressive than the first one and may be harder to perform.

3.2.5 Scenario 4: Public Sybil attack by disabling nodes

Once again, the difference between the public attack and the localized one depends on the number of neighbors the compromised node wants to send different identities to.

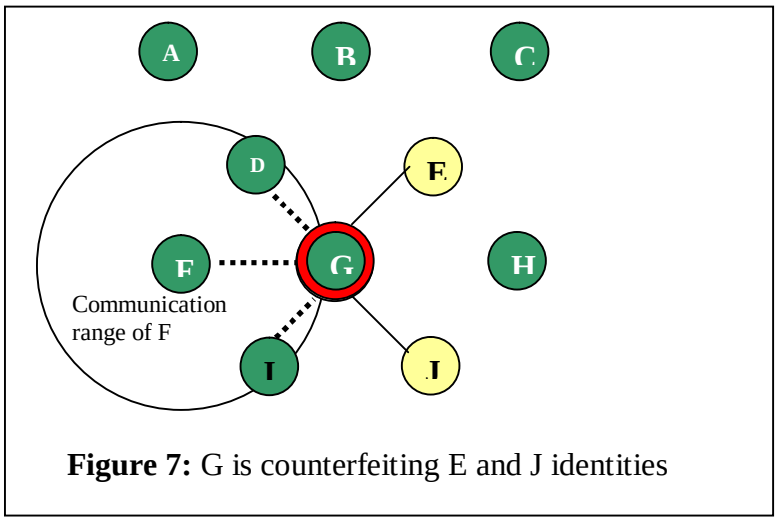


Figure 7: G is counterfeiting E and J identities

3.3 Sybil Attack Detection Models

Here several methods have been proposed to protect Sybil attacks in ad hoc networks.
The protocols used for *secured ad hoc networks* can be broadly classified in three groups as follows:

a) PKI based protocols
One way to prevent Sybil attack is to *maintain a central identification authority* to confirm each identity along with its identity to others. A central authority is required to distribute cryptographic material prior to or during deployment. However, no perfect implementation technique for this protocol exists and the less flexible nature of this central authority scheme is not desirable in ad hoc communications. Allowing nodes to join without pre-distribution keys leaves a potential probability for Sybil attack.

b) Threshold based protocols
In this scheme a group of trusted nodes distributes cryptographic material only if new members *satisfy some threshold criteria*. This method does not provide high level security because Sybil attacker can take control of the network by generating the identities to meet the threshold requirements.

c) Reputation schemes
Reputation schemes include protocols *determining and maintaining information* regarding the trustworthiness about the nodes in the group. However, the Sybil attack undermines these protocols because a node can use multiple identities to falsify vouch for or otherwise support an identity that would otherwise gain a bad reputation.

3.4 Approaches to detect some attacks

3.4.1 Watchdog Mechanism

Watchdog mechanism is implemented in Dynamic Source Routing. When a node sends a packet, it sees if the next node properly forwards it to the other node within some given internal of time. The buffer of the node maintains the record of the recently send packet and erase it on successful transmission by the neighboring node. If not transmitted within threshold time interval, it is suspected as a misbehaving node. However, this method may be ineffective in case when the malicious node with several IDs pretends as if it is forwarding the message by intentionally broadcasting the message.

3.4.2 Improved Watchdog – like Mechanism

This method is based on the host in cooperation with the neighbors. The method relies on the acknowledgement broadcasted by the destination node and received by its neighbors. In case the compromised node sends a false forwarding message and no response signal message is observed from the destination, the neighbors of the destination machine detects the compromised host and informs the original sender.
The neighboring nodes can inform the original sender in two ways: either they inform after observing the receipt notification from the destination which is referred as Passive Reaction Policy, or they may inform the original source when no acknowledgement from the destination is observed which is known as Active Reaction Policy. Limitations are if the compromised node broadcasts the receipt notification, the attack cannot be detected and in case of edge nodes and nodes in the sparse network there may be no neighboring nodes and hence this method cannot be applied.

3.4.3 Anomaly detection features

This mechanism is leads to determination of clustering of normal activity versus Sybil attack by using features like routing protocol and certain features of Watchdog mechanism. Anomaly detection is based on the parameters like message received, message forwarded, through message, neighbor list, number of through destination. Also similar to above but this technique differs from them in the respect that some threshold value has been used for anomaly detection. The anomaly detection module is trained and tested on a set of normal and abnormal data. Each node is entrusted with the responsibility of monitoring the selected audit features and identifying the abnormally high confidence associated with the neighbors.
The main drawback of this method is that it assumes that the audit features can detect Sybil attack which is an unknown factor at present. Moreover, features depend on the routing protocols and training phase overhead requires to be justified. Also an effective classification model should be chosen to make the method efficient.

3.4.4 Collaboration of misuse Watchdog in neighbors

The limitations of the aforesaid techniques can be alleviated through exchanging information among the "watchdog" components of neighbors. Exploiting the information provided by the neighboring nodes, a picture of local neighborhood can be visualized. In case the results differ beyond a predetermined margin, the attacker is suspected. The neighbors vote on any result only after completion of exchange of information. The attacker is detected with high confidence according to the result of voting.
However, overhead is more in this technique and voting should be done carefully.

3.4.5 Extended Watchdog misuse feature

The detection rate and the robustness of the existing Watchdog mechanism can be increased by incorporation of two new features: route load count and routing forwarding confusion detection mechanism.
This scheme is based on the assumption of certain topology and existence of a method to determine overuse of a node as compared to high legitimate usage of a node and hence does not provide any guarantee for the success of the scheme.

3.4.6 Neighboring Nodes confirmation

This scheme is based on the fact that two different nodes should have different neighbors. In this mechanism a node asks other nodes to inform about their neighbors. If two nodes are found to have same neighbors, it can be concluded that those nodes are the same and may be the Sybil attacker.
Limitations are if the malicious node sends forged information about their neighbors, it cannot be detected and if two legitimate nodes which are very close, may have the same neighbors and may be falsely interpreted as compromised node by this technique.

So our approach for the detection of Sybil attack in our network will be by neighboring node confirmation or by using neighboring information. More about how the common neighbors are found and detection mechanism will be talked in depth in the next topic.

CHAPTER 4

DETECTION MECHANISM

4.1 Detection of Sybil Attack in a MANET Using Neighboring Information

Sensor nodes are resource constrained in terms of

- *Memory capacity*
- *Computational abilities*

Each node communicates via wireless channel with neighboring nodes.

In such network there are high risks of open attacks by malicious eavesdropper, thus major concern if the data is of critical nature or important data.

Sybil attacks pose a serious threat to the integrity of a network. In such attack, a single malicious node forges multiple entities within a network in order to mislead the genuine node into believing that they have many neighbors.

Assumptions that the probability of two nodes having exactly same set of neighbors is extremely low provided that the network has a high node density that is the numbers of node that are willing to communicate with each other and formed the ad hoc network.

In Sybil attack, the forged nodes typically have the same sets of neighbors because they are associated with the same physical device i.e. the malicious node which has created the Sybil nodes.

4.2 Assumption and attack model

Then senor nodes n are randomly distributed in some m x m matrix. Nodes are taken to be stationary and unaware of their location. Communication is done via wireless radio channel and broadcast in omni-direction mode. Node transmits a message and the message is received by only those nodes within the communication range. (designated as “neighboring nodes”).

Compromised node is known as malicious node and rest of node in the ad hoc will be normal nodes. Malicious cheats its neighbors by creating multiple identities referred as Sybil nodes. When a node sends a message to Sybil nodes, the message is received and replied by malicious node if needed.

Malicious node main purpose is to cheat/trick the normal node in the network into believing that they are having many neighbors for their disposal which is untrue.

It also assumed that the number of Sybil nodes is greater than normal nodes amongst the neighboring set in the ad hoc network.

4.2.1 Detecting of Sybil nodes

When a malicious node discerns itself to be within the communication range of a normal node, it immediately forges multiple Sybil nodes. Sybil nodes being associated with same physical node (i.e. malicious node), they share the same set of neighbors. Detection of Sybil node can be done by collecting neighboring information and analyzing the result in hand.

There should be a set of nodes whose appearance times are higher. The objective of the detection mechanism is to find such set, called critical set, C, and determine the Sybil nodes by the assistance of such set.

4.2.2 Corresponding notations to be taken:

SN: The set of all of the sensor nodes, including Sybil nodes forged by malicious nodes.

M: The malicious node responsible for the Sybil attack.

$\mathbf{NB_i}$: The set of i's neighbors, $\mathbf{i \in SN}$.

$\mathbf{CNB_{i,j}}$: The set of common neighbors for both i and j, $\mathbf{i, j \in SN, i \neq j.}$
Therefore, $\mathbf{CNB_{i,j} = NB_i \cap NB_j.}$

$\mathbf{NB^n_i}$: The set of i's normal neighbors, $\mathbf{i \in SN}$. Since $\mathbf{NB_i}$ contains all of i's neighbors, including Sybil nodes and normal nodes, $\mathbf{NB^n_i \subset NB_i}$.

$\mathbf{CNB^n_{i,j}}$: The set of common normal neighbors for both i and j.
Also, $\mathbf{CNB^n_{i,j} \subset CNB_{i,j}.}$

Protection scheme is executed by the normal node which suspect itself to be a ***victim of a Sybil attack***, such node are called ***V.***

4.2.3 To determine each $CNB_{i,v}$:

1. Node V broadcasts a request message to one of its neighbor. e.g. node i
2. When i receives the message, it broadcast it over maximum transmission range
3. Any node hearing the message from i (e.g. node j) replies using one hop broadcast directly to **V**
4. Node v records the IDs of nodes which send a reply and combines the IDs to form the set $\mathbf{CNB_{i,v}}$
5. Repeat the above until all the V $\mathbf{CNB_{i,v}}$ have been collected.

Sybil node detection method is based on a simple information collection and analysis approach rather than direct interrogation of each node.

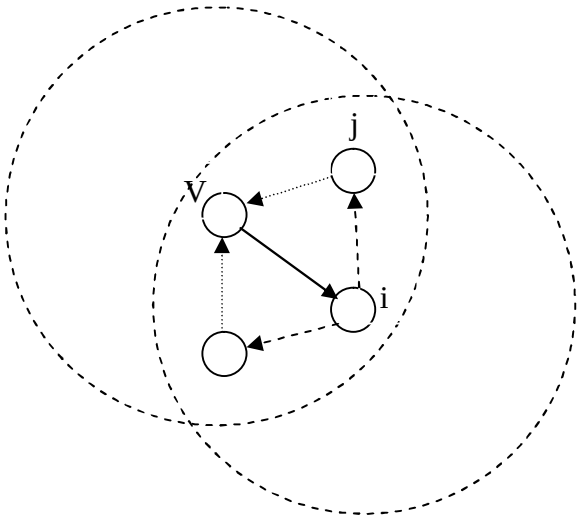


Figure 8: Node V collecting $CNB_{i,v}$

- When i broadcast its message, it is heard not only by its neighbors but also by **V** itself. If **V** doesn't observe such reply, then i will be treated as a Sybil/ malicious node.
- After information is collected, V will then count the numbers of time each node ID appears in the $\mathbf{CNB_{i,v}}$ complied by **V**.
- AP(i): total number of appearances of node i in $\mathbf{CNB_{i,v}}$**(for all j ϵ** $\mathbf{NB_v}$**)**

- Sybil nodes have same set of neighbors and amount of Sybil nodes are more than normal nodes, there will be some nodes whose **AP(i)** is much higher and these nodes are consequently neighbors of Sybil nodes.

- Critical member, $\mathbf{C=\{ i \mid AP(i) > \theta , i \in NB_v \}}$

  θ is the threshold value set with proper setting.

- The threshold parameter θ:

  θ=0.6|N| to 0.85|N| where |N| is total number of V's neighbors.

- Node i is considered as a Sybil node if $\mathbf{CNB_{i,v} \supset C}$

4.2.4 Procedure for detecting Sybil nodes in a MANET

1. Let node density be d and the number of nodes in a m x m matrix is n.
2. Assignment of neighbors to the nodes. It totally depends on the communication range of the node.
3. Let any node be victim node V which suspects a Sybil attack.
4. Now node V will broadcast to its neighboring nodes to find their neighboring nodes in their vicinity such that their common neighbors could be found out.
5. Step 3 is repeated until all the neighbors of V is covered.
6. Then the Appearance of each node is calculated.
7. A threshold value is assigned so as we can determine the critical set, C.
8. Only those nodes whose appearance is higher than the threshold value is taken as critical set/member.
9. Any node that include the entire critical members in their common neighbor set is taken as Sybil node
10. As per needed, enhancement mechanism can be used if any nodes are erroneously identified as Sybil node if it contain the complete set C. (False detection result)

4.3 Algorithm for detecting Sybil nodes in a MANET

Calculation of $CNB_{i,v}$ and A_i

CNB_A(node[],V)

Neighbor [V]=calculate_neighbor[V]

V sends request to each node in neighbor[V]

Neighbor[V] sends request to its farthest transmission range

For each neighbor of V

Acknowledgement[V,i]=Acknowledgement Received

$CNB_{i,v}$ =Intersection(Neighbor[V] , Acknowledgement[V,i])

For each entry in $CNB_{i,v}$ calculate A_i

Calculation of Critical Set

CS(A)

Critical_value =0.7*number of nodes

Critical_set=NULL

For each entry in A

if(A_i > Critical_value)

Critical_set=Critical_set U A_i

Detection of Malicious node or Sybil nodes

M_nodes()

For each neighbor[V]

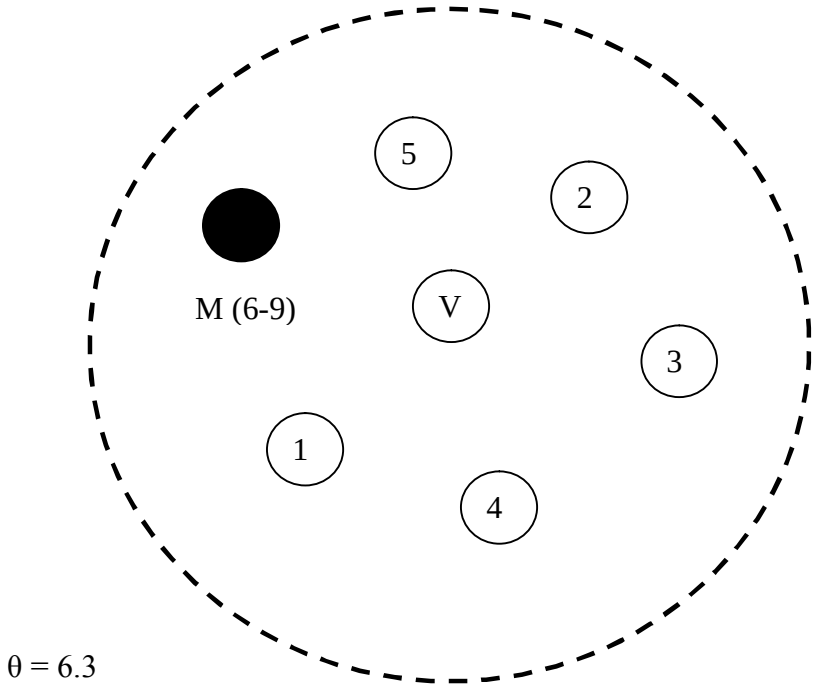
if($CNB_{i,v}$ n Critical_set==Critical_set)

Node i is malicious

else

Not malicious node

ID	Times
1	7
2	3
3	2
4	3
5	7
6	4
7	4
8	5
9	5



$CNB_{1,V}$={4,5,7,8}
$CNB_{2,V}$={1,4,3,8,9}
$CNB_{3,V}$={1,2,7}
$CNB_{4,V}$={1,2,5,6,8}
$CNB_{5,V}$={2,3,4,6}
$CNB_{6,V}$={1,5,8,9}
$CNB_{7,V}$={1,5,8,9}
$CNB_{8,V}$={1,5,6,7,9}
$CNB_{9,V}$={1,5,6,7}

Figure 9: Illustration of detection method

In the above figure, node 6 to 9 assumed as Sybil nodes forged by malicious node M. Nodes 1 to 5 are normal nodes.

The critical value, θ is set to 6.3 so the critical member, C= {1,5}.

So nodes 6 to 9 include the complete set of C and is taken as Sybil node.

Node 4 is wrongly identified as Sybil node as it includes critical set. This is fault detection. This fault detection result can be reduced by using the enhancement technique/method.

4.4 Flow chart

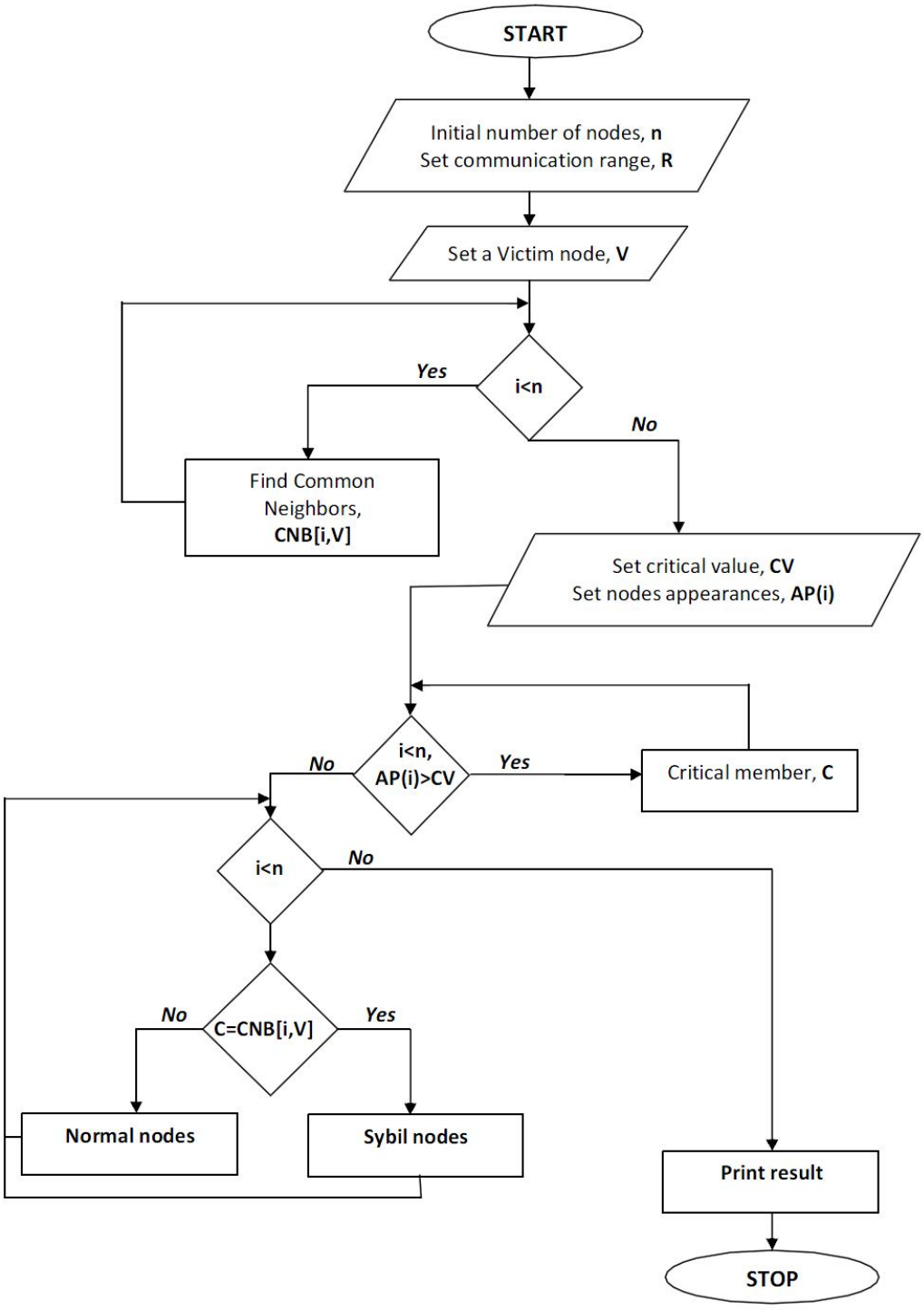


Figure 10: Flow chart for detecting Sybil nodes

4.5 Enhancement Mechanism

When malicious node M is far from node V, number of common normal neighbor of V and malicious node is less which means then critical set is smaller. When the size of critical set is smaller, the probability that a normal node has critical set in its neighbor set is higher. Therefore there will be more false positive.

This mechanism talks about excluding M such that it doesn't have effect on node V and its neighbors will be all normal nodes. This is done by adjusting the communication range of V. The communication range of V is reduced until the malicious node falls outside of the range such that M has no longer any effect. Thus by doing this, the false detection is reduced.

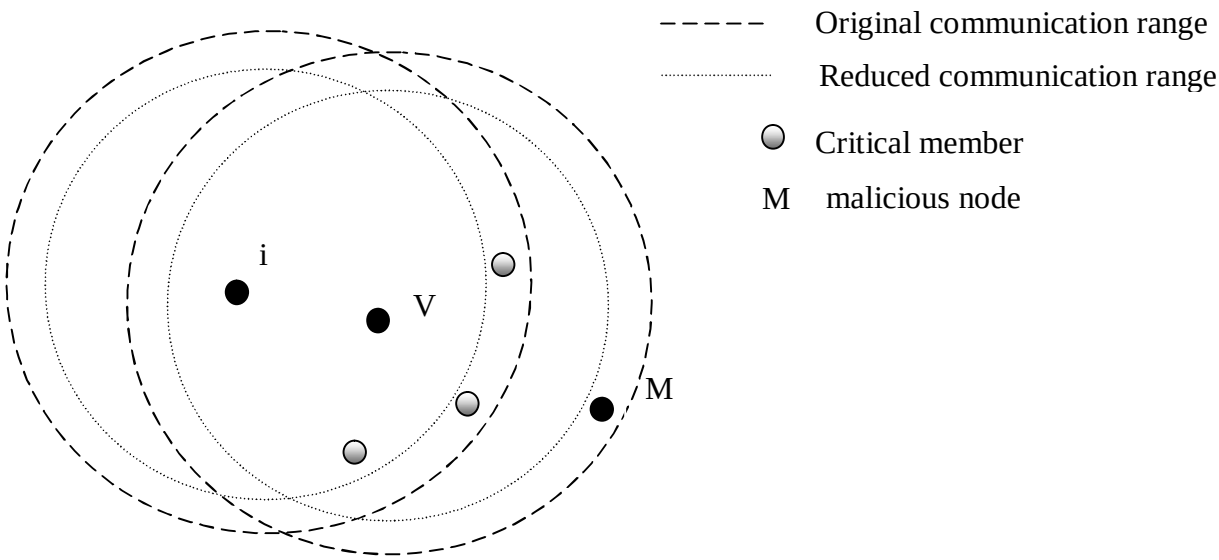


Figure 11: Enhance detection mechanism

After the detection procedure described above, V possesses a set S whose members represents the Sybil nodes by V. V reduces communication range until number of S is half less than previous one, as it can be inferred now as malicious node M is out of V's communication range and enhancement method can be stopped.

CHAPTER 5
IMPLEMENTATION

5.1 Network Simulator - ns2 (In 1989)

Open-source and event driven simulator

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using

NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviours.

Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. University of California and Cornell University is the one who developed the REAL network simulator, the foundation on which NS is based on. Since 1995 the Defence Advanced Research Projects Agency (DARPA) supported development of NS through the Virtual InterNetwork Testbed (VINT) project. Currently the National Science Foundation (NSF) has joined the ride in development. Last but not the least, the group of researchers and developers in the community are constantly working to keep NS2 strong and versatile.

Installation guidelines are given in the Appendix A.

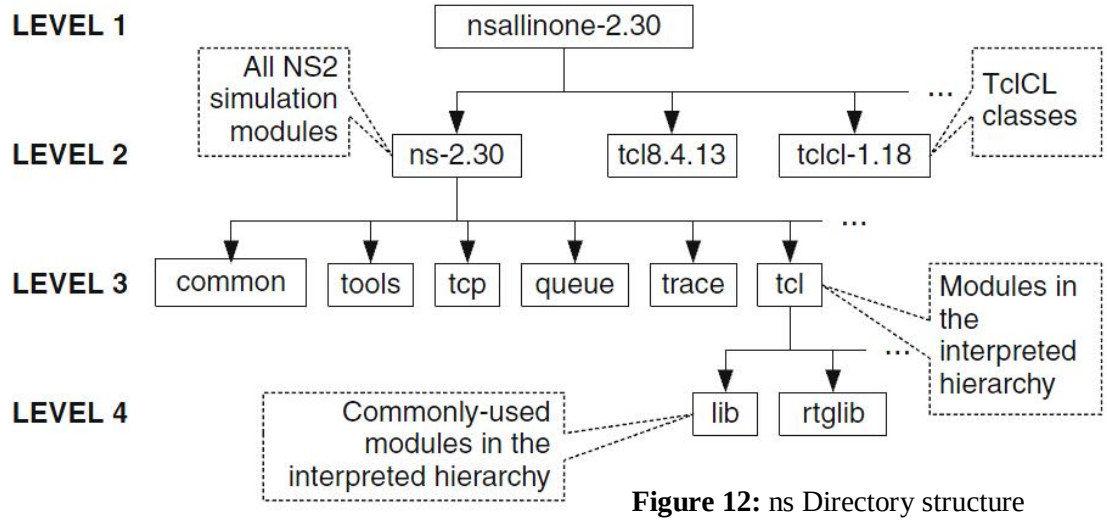


Figure 12: ns Directory structure

5.1.1 Architecture of NS2

NS2 provides users with an executable command 'ns' which takes on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns. In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation.

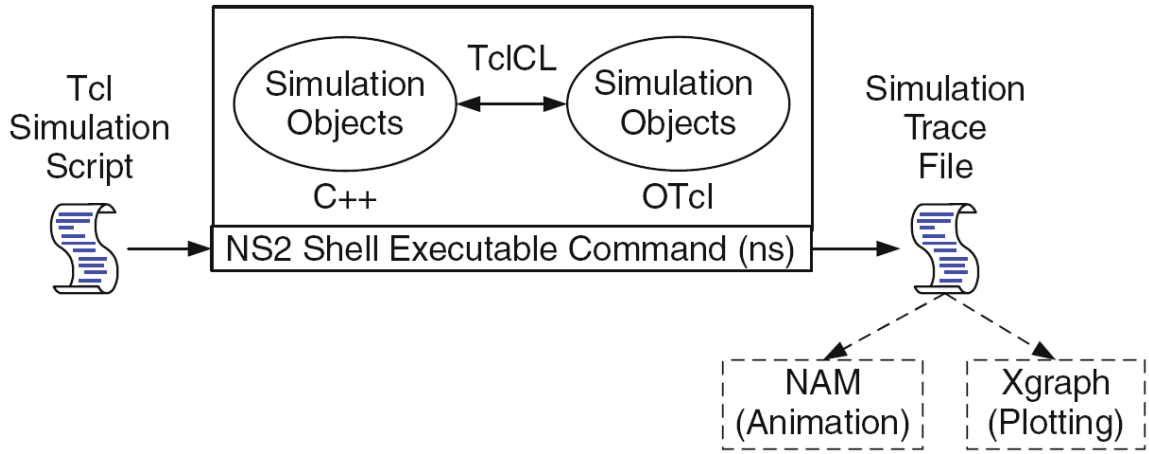


Figure 13: Basic architecture of NS

NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as *handle*s.

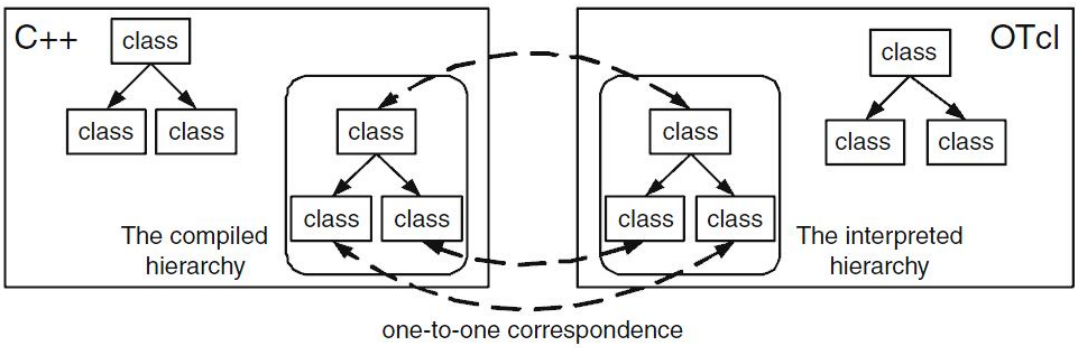


Figure 14: OTcl and C++: The Duality

Conceptually, a handle (e.g., n as a Node handle) is just a string (e.g.,_o10) in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector).

In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may define its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively.

NS2 provides a large number of built-in C++ objects and it is advisable to use these C++ objects to set up a simulation using a Tcl simulation script. After simulation, NS2 outputs either text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used.

Simple code linkage to architecture of ns2:

Step 1:

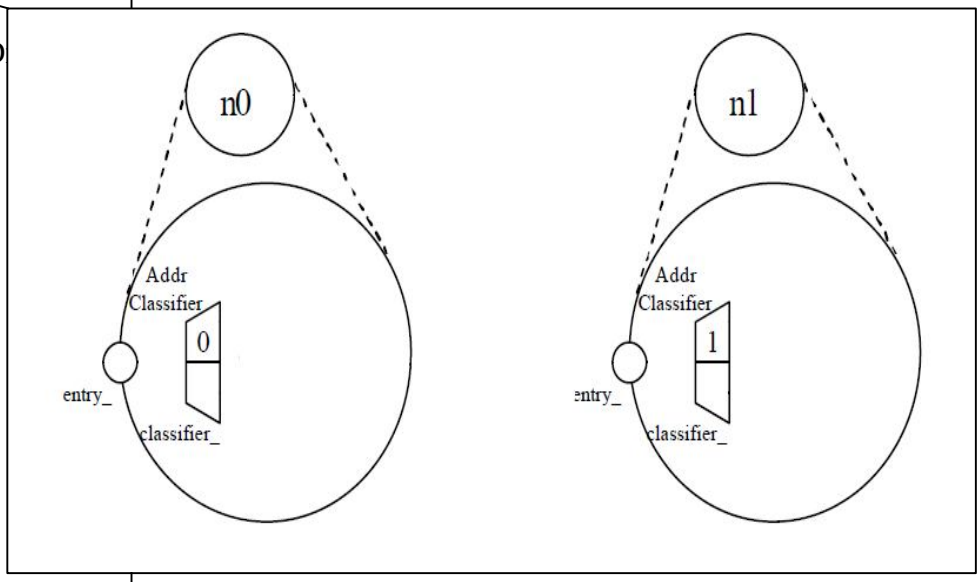
```
set ns [new Simulator]

set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n1 $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns at 0.5 "$ftp0 start"
$ns at 4.5 "$ftp0 stop"
$ns at 5.0 "stop"

$ns run                    #simplecode.tcl
```

```
Simulator instproc init args {
        $self create_packetformat

        $self use-scheduler Calendar

        $self set nullAgent_  [new Agent …]
        $self set_address_format def….

        Eval $self next $args
}
```

Run Program:
%ns simplecode.tcl

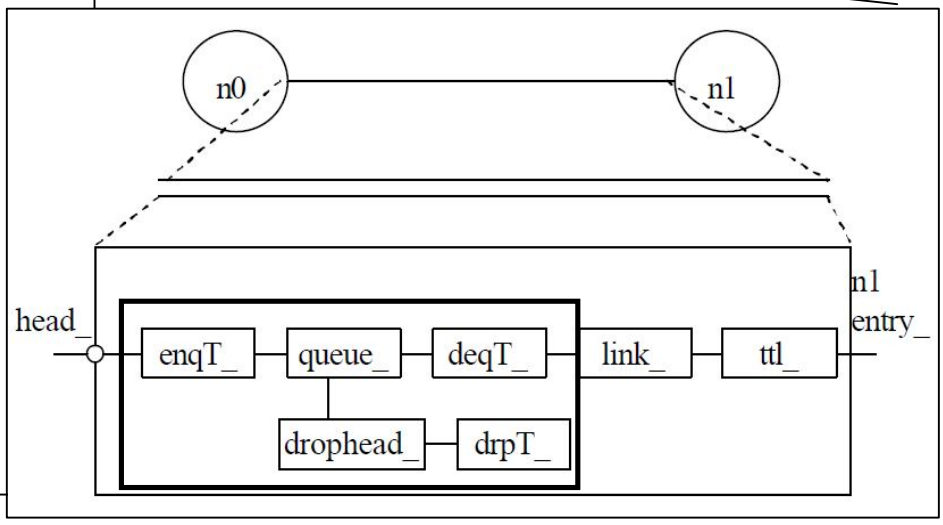
Step 2:

```
set ns [new Simulator]

set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms D

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n1 $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns at 0.5 “$ftp0 start”
$ns at 4.5 “$ftp0 stop”
$ns at 5.0 “stop”

$ns run
```



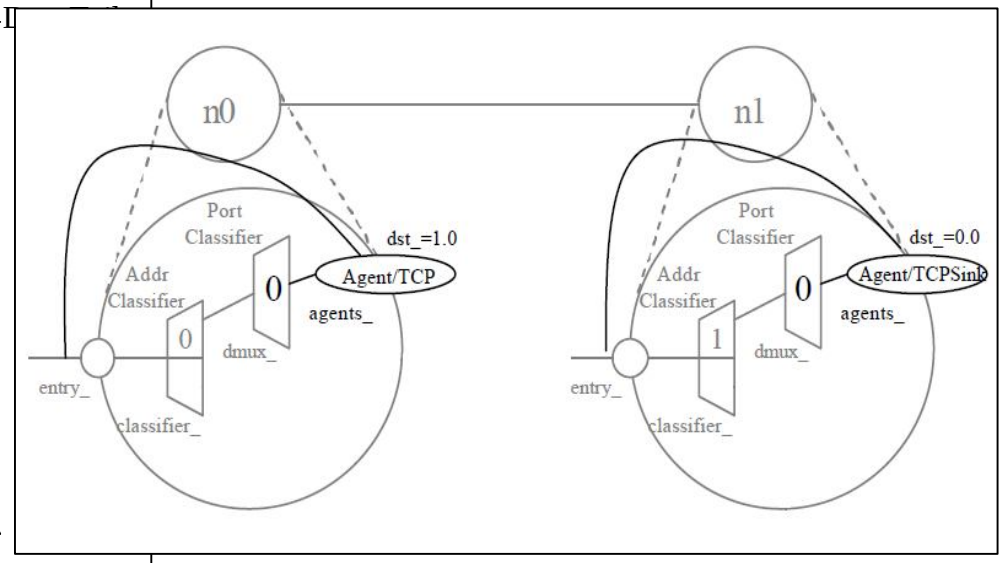
Step 3:

```
set ns [new Simulator]

set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n1 $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns at 0.5 “$ftp0 start”
$ns at 4.5 “$ftp0 stop”
$ns at 5.0 “stop”

$ns run
```



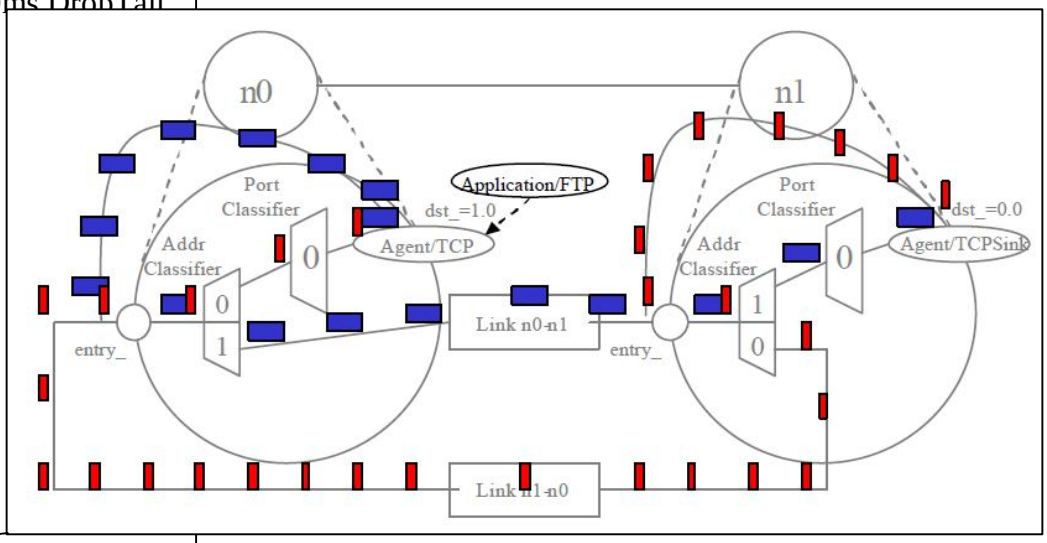
Step 4:

```
set ns [new Simulator]

set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms D

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n1 $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns at 0.5 "$ftp0 start"
$ns at 4.5 "$ftp0 stop"
$ns at 5.0 "stop"

$ns run
```



Step 5:

```
set ns [new Simulator]

set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n1 $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns at 0.5 "$ftp0 start"
$ns at 4.5 "$ftp0 stop"
$ns at 5.0 "stop"

$ns run
```



5.1.2 Visualization the Network flow

To visualize the flow of packet between the node the "Nam Editor" also known as Network Animator is used. It has option of zooming, resetting the layout of the architecture designed through tcl script. It has start, stop, forward, fast forward, rewind and fast rewind options which control the flow of simulation time.

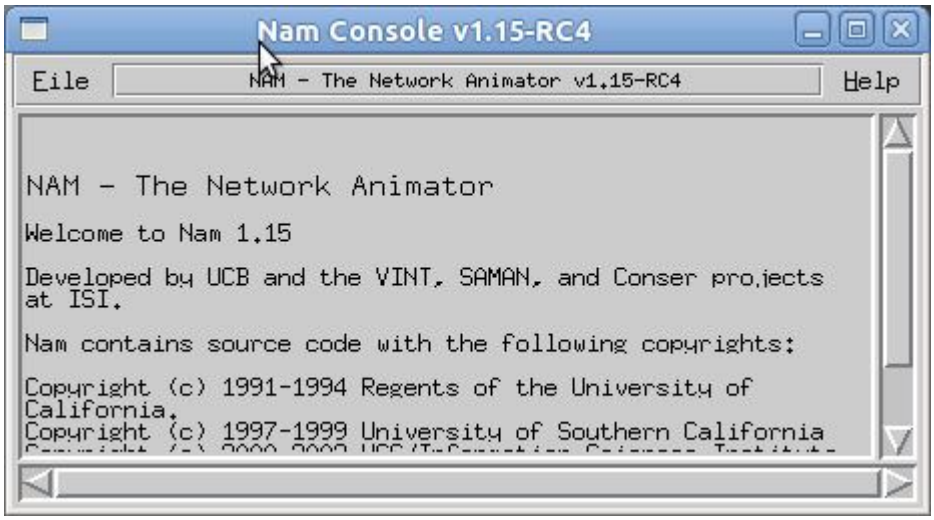


Figure 15: Nam Console

5.1.3 Analyzing: Trace File Format

When the ns code is running, the trace of each event is stored in a trace file. While tracing into an output ASCII file, the trace is organized in 12 field as shown in the following figure. The description of each field is shown by their name and an sample example of the trace file is also given.

Type Identifier	Time	Source Node	Destination Node	Packet Name	Packet Size	Flags	Flow ID	Source Address	Destination Address	Sequence Number	Packet Unique ID

r : receive (at to node)
+ : enqueue (at queue)
- : dequeue (at queue)
d : drop (at queue)

src addr : node.port (1.0)
dst addr : node.port (0.0)

```
+ 0.110419 1 2 tcp 1040 ------- 2 1.0 4.0 5 12
+ 0.110419 1 2 tcp 1040 ------- 2 1.0 4.0 6 13
- 0.110431 1 2 tcp 1040 ------- 2 1.0 4.0 5 12
- 0.110514 1 2 tcp 1040 ------- 2 1.0 4.0 6 13
r 0.11308 0 2 cbr 1000 ------- 1 0.0 3.0 2 8
+ 0.11308 2 3 cbr 1000 ------- 1 0.0 3.0 2 8
- 0.11308 2 3 cbr 1000 ------- 1 0.0 3.0 2 8
r 0.11316 0 2 cbr 1000 ------- 1 0.0 3.0 3 9
```


Figure 16: Trace file structure

5.2 Programming

5.2.1 NS2 Code

```
#filename: pro.tcl
#setting adhoc network
set val(chan)          Channel/WirelessChannel    ;# channel type
set val(prop)          Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)         Phy/WirelessPhy            ;# network interface type
set val(mac)           Mac/802_11                 ;# MAC type
set val(ifq)           Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)            LL                         ;# link layer type
set val(ant)           Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)        50                         ;# max packet in ifq
set val(nn)            15                         ;# number of mobilenodes
set val(rp)            AODV                       ;# routing protocol
set val(x)                                        350
set val(y)                                        350
set val(stop)    150                                                    ;


set ns [new Simulator]
set tracefile      [open trace_pro.tr w]
set nam [open nam_pro.nam w]

$ns trace-all $tracefile
$ns namtrace-all-wireless $nam $val(x) $val(y)

set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)

set chan [new $val(chan)]

        $ns node-config -adhocRouting $val(rp) \
                        -llType $val(ll) \
                        -macType $val(mac) \
                        -ifqType $val(ifq) \
                        -ifqLen $val(ifqlen) \
                        -antType $val(ant) \
                        -propType $val(prop) \
```

```
                        -phyType $val(netif) \
                        -channel $chan \
                        -topoInstance $topo \
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace OFF \
                        -movementTrace OFF


        for {set i 0} {$i < $val(nn) } {incr i} {
                set node_($i) [$ns node]
                $node_($i) random-motion 0              ;# disable random motion
        }

#Setting the nodes initial point

$node_(0) set X_ 5.0
$node_(0) set Y_ 2.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 100.0
$node_(1) set Y_ 100.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 290.0
$node_(2) set Y_ 285.0
$node_(2) set Z_ 0.0

$node_(3) set X_ 190.0
$node_(3) set Y_ 185.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 0.0
$node_(4) set Y_ 20.0
$node_(4) set Z_ .0

$node_(5) set X_ 200.0
$node_(5) set Y_ 155.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 100.0
$node_(6) set Y_ 185.0
$node_(6) set Z_ 0.0
```

```
$node_(7) set X_ 190.0
$node_(7) set Y_ 105.0
$node_(7) set Z_ 0.0

$node_(8) set X_ 90.0
$node_(8) set Y_ 150.0
$node_(8) set Z_ 0.0

$node_(9) set X_ 200.0
$node_(9) set Y_ 175.0
$node_(9) set Z_ 0.0

#Adding motion to the node i.e setting direction and speed

$ns at 7.0 "$node_(3) setdest 110.0 100.0 35.0"
$ns at 7.0 "$node_(2) setdest 150.0 250.0 35.0"
$ns at 1.0 "$node_(1) setdest 325.0 50.0 35.0"
$ns at 7.0 "$node_(0) setdest 10.0 10.0 1.0"

$ns at 7.0 "$node_(4) setdest 80.0 60.0 15.0"
$ns at 7.0 "$node_(5) setdest 120.0 250.0 25.0"
$ns at 1.0 "$node_(6) setdest 200.0 10.0 9.0"

$ns at 7.0 "$node_(7) setdest 200.0 150.0 6.0"

$ns at 7.0 "$node_(8) setdest 170.0 150.0 20.0"
$ns at 7.0 "$node_(9) setdest 120.0 50.0 22.0"

        for {set i 10} {$i < $val(nn)-2 } {incr i} {
                $node_($i) set X_ [expr 100.0+($i*10.0)]
                $node_($i) set Y_ [expr 100.0+($i*10.0)]
                $node_($i) set Z_ 0.0
        }

$node_(13) set X_ 200.0
$node_(13) set Y_ 220.0
$node_(13) set Z_ 0.0

$node_(14) set X_ 220.0
$node_(14) set Y_ 200.0
$node_(14) set Z_ 0.0


        for {set i 10} {$i < $val(nn)-2 } {incr i} {
```

```
                $ns at 7.0 "$node_($i) setdest [expr (150.0+10.0*$i)] [expr
(50.0+10.0*$i)] 35.0"
        }


$ns at 7.0 "$node_(13) setdest 250.0 170.0 35.0"
$ns at 7.0 "$node_(14) setdest 270.0 150.0 35.0"

#Transfer of packet from one node to another node

set tcp1 [new Agent/TCP]
$tcp1 set class_ 1
set sink1 [new Agent/TCPSink]
$ns attach-agent $node_(0) $sink1
$ns attach-agent $node_(1) $tcp1
$ns connect $tcp1 $sink1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 15.0 "$ftp1 start"

set tcp2 [new Agent/TCP]
$tcp2 set class_ 2
set sink2 [new Agent/TCPSink]
$ns attach-agent $node_(1) $sink2
$ns attach-agent $node_(2) $tcp2
$ns connect $tcp2 $sink2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ns at 20.0 "$ftp2 start"

set tcp3 [new Agent/TCP]
$tcp3 set class_ 3
set sink3 [new Agent/TCPSink]
$ns attach-agent $node_(5) $sink3
$ns attach-agent $node_(1) $tcp3
$ns connect $tcp3 $sink3
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
$ns at 25.0 "$ftp3 start"

set tcp4 [new Agent/TCP]
$tcp4 set class_ 4
set sink4 [new Agent/TCPSink]
$ns attach-agent $node_(9) $sink4
```

```
$ns attach-agent $node_(2) $tcp4
$ns connect $tcp4 $sink4
set ftp4 [new Application/FTP]
$ftp4 attach-agent $tcp4
$ns at 25.0 "$ftp4 start"

#data passing with sybil node

set tcp5 [new Agent/TCP]
$tcp5 set class_ 5
set sink5 [new Agent/TCPSink]
$ns attach-agent $node_(13) $sink5
$ns attach-agent $node_(3) $tcp5
$ns connect $tcp5 $sink5
set ftp5 [new Application/FTP]
$ftp5 attach-agent $tcp5
$ns at 20.0 "$ftp5 start"

proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file"
$ns at 10.1 "plotWindow $tcp4 $windowVsTime2" }

for {set i 0} {$i < $val(nn) } {incr i} {
   $ns at 150.0 "$node_($i) reset";
}

$ns at 150.0 "stop"
$ns at 150.01 "puts \"NS EXITING...\" ; $ns halt"

#STOP the simulation and add the files in their respective directory
proc stop {} {
   global ns tracefile nam
   $ns flush-trace
   close $tracefile
   close $nam
}
```

```
puts "Starting Simulation..."
puts "Mobile Adhoc Network"
$ns run
```

5.2.2 Screenshot NAM

Figure 17: Wireless network

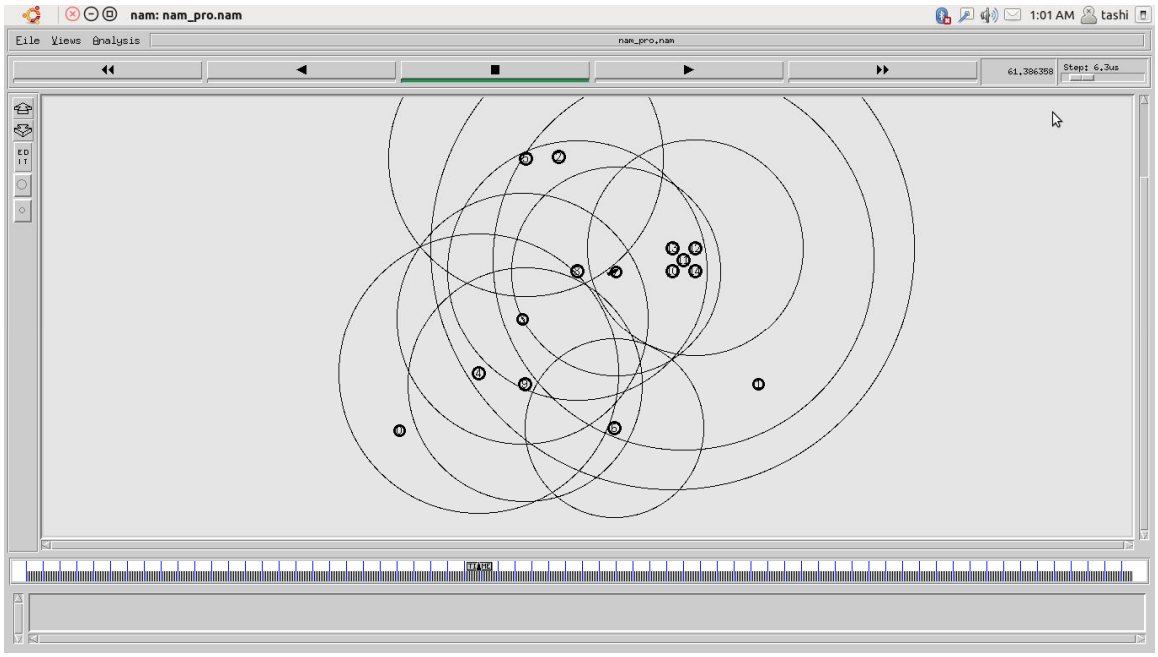


Figure 18: Wireless network: Computing nearest neighbor

Figure 19: Packet flow

5.2.3 C++ code:

```
//filename: sybil.cpp
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#include<dos.h>

int network[100][100],neighbor[50][25],cnv[25][25],ap[50],critical[50],sybil[40];

int i,r,j,k,l,m=0,v,max_X,max_Y,nodes_T,nodes_M,cv,ch;
int x,y2,y,x2;
long double x1,y1,fac;

int length(int a[])
{
  int i=0;

  while(a[i])
    i++;
  return i;
}
```

```
long double fact(int a)
{
fac=1;
  while(a!=0)
        {
         fac=fac*a;
         a--;
        }
  return fac;
}

void sybil_nodes()
{
 int i,j,k,f=0,t=0,c=0;

  for(i=0;i<=nodes_T+nodes_M;i++)
    {
      for(k=0;k<length(critical);k++)
          {
          t=0;
          for(j=0;j<length(neighbor[i])&&!t;j++)
          {
            if(critical[k]==neighbor[i][j])
                {
                        t=1;
                        c++;
                        delay(10);
                }
            }
          }

          if(c==length(critical))
          {
             sybil[f++]=i;
             c=0;
          }
     }

    cout<<"\n\nsybil nodes are =";
  for(i=0;i<length(sybil);i++)
  {
   cout<<" "<<sybil[i];
  }
}
```

```
void calculate_ap()
{
int i,j,k=0;

  for(i=0;i<length(neighbor[v]);i++)
    for(j=0;j<length(neighbor[neighbor[v][i]]);j++)
     {
       ap[cnv[i][j]]=1+ap[cnv[i][j]];
     }

  k=0;

 for(i=1;i<=nodes_T+nodes_M;i++)
    {
     if(ap[i]>cv)
      {
       critical[k++]=i;
      }
    }
  cout<<"\nCritical set::";

  for(i=0;i<length(critical);i++)
    {
      cout<<critical[i]<<" ";
    }
}

void cnb_IV()
{
 int m,i,j,k=0,t,s;

 for(m=0;m<length(neighbor[v]);m++)
    {
    k=0;
    for(i=0;i<length(neighbor[v]);i++)
      for(j=0;j<length(neighbor[neighbor[v][m]]);j++)
         {
            if(neighbor[v][i]==neighbor[neighbor[v][m]][j])
                    {
                        cnv[m][k++]=neighbor[v][i];
                    }
          }
     }
}
```

```
void calculate_neighbor()
{
  int x,y,i,j,k=0,p,q=-1,f,d;

  for(x=0;x<max_X;x++)
    for(y=0;y<max_Y;y++)
          if(network[x][y]!=0)
            {
                    if(network[x][y]!=nodes_T+1)
                      {
                            k=0;
                            for(i=x-r;i<=x+r;i++)
                              for(j=y-r;j<=y+r;j++)
                                    {
                                    if(network[i][j]!=0&&network[x][y]!=network[i][j])
                                      {
                                            if(network[i][j]!=nodes_T+1)
                                               {
                                                neighbor[network[x][y]][k]=network[i][j];
                                                k++;
                                                      if(m<k)
                                                       m=k;
                                                }
                                               else
                                                {
                                                      for(p=nodes_T+1;p<=(nodes_T+nodes_M);p++)
                                                       {
                                                              neighbor[network[x][y]][k]=p;
                                                              k++;
                                                       }
                                                         if(m<k)
                                                          m=k;
                                                 }
                                        }

                                    }
                       }

                    else
                       {
                            for(i=x-r;i<=x+r;i++)
                              for(j=y-r;j<=y+r;j++)
                                 {
                                  if(network[i][j]!=0)
```

```
                                {
                                    q++;
                                    for(p=nodes_T+1;p<=(nodes_T+nodes_M);p++)
                                      {
                                      neighbor[p][q]=network[i][j];
                                        }
                                  }

                                }
                              f=q;

for(q=q,d=nodes_T+1;q<f+nodes_M&&d<=nodes_T+nodes_M;q++,d++)
                              for(p=nodes_T+1;p<=(nodes_T+nodes_M);p++)
                                {
                                    neighbor[p][q+1]=d+1;
                                }

                          }
                }

     if(m<q)
      m=q;
    cout<<"\nNeighbor of all nodes\n";

   for(i=1;i<=nodes_T+nodes_M;i++)
     { cout<<"\n";
    for(k=0;k<m;k++)
    cout<<neighbor[i][k]<<" ";
     }

}


void create_network()
{
  cout<<"Enter the size of network area\nmax_X=:";
  cin>>max_X;
  cout<<"\nmax_Y=";
  cin>>max_Y;
  cout<<"\nEnter the valid nodes in a network nodes_T=:";
  cin>>nodes_T;
  cout<<"\nEnter the total malicious nodes in a network nodes_M=:";
  cin>>nodes_M;
  cout<<"Enter the node range:";
```

```
  cin>>r;
cout<<"enter the choice of distribution";
  cin>>ch;

  for(i=0;i<100;i++)
    for(j=0;j<100;j++)
      network[i][j]=0;
   k=0;

switch(ch)
  {
  case 1:
  for(i=0;k<(nodes_T);i++)
          {

            x=r+rand()%(max_X-(2*r));
            y=r+rand()%(max_Y-(2*r));
            if(network[x][y]==0)
             {
               network[x][y]=k+1;
               k++;
             }
          }
          k=1;
      while(k)
      {
           x=r+rand()%(max_X-(r*2));
           y=r+rand()%(max_Y-(r*2));
           if(network[x][y]==0)
            {
             network[x][y]=nodes_T+1;
             x=r+rand()%(max_X-(2*r));
             y=r+rand()%(max_Y-(2*r));
             k=0;
            }
      }
break;

     case 2:
       x1=((exp(max_X))*(pow(max_X,r))/(fact(r)));
       x2=int(x1);
       cout<<x2;
       y1=((exp(max_Y))*(pow(max_Y,r))/(fact(r)));
       y2=int(y1);
```

```
cout<<y2;
for(i=0;k<(nodes_T);i++)
    {
     x=r+rand()%(x2);
     y=r+rand()%(y2);
     if(network[x][y]==0)
      {
        network[x][y]=k+1;
        k++;
      }
    }
    k=1;
 while(k)
 {
     x=r+rand()%(x2);
     y=r+rand()%(y2);
     if(network[x][y]==0)
      {

       network[x][y]=nodes_T+1;
       x=r+rand()%(x2);
       y=r+rand()%(y2);
       k=0;
      }
 }
     cout<<"pls";
 break;

case 3:
 x1=((exp(max_X*r))*r);
 x2=int(x1);
 cout<<x2;
 y1=((exp(max_Y*r))*r);
 y2=int(y1);
cout<<y2;
for(i=0;k<(nodes_T);i++)
    {
     x=r+rand()%(x2);
     y=r+rand()%(y2);
     if(network[x][y]==0)
      {
        network[x][y]=k+1;
        k++;
      }
```

```
        }
        k=1;
      while(k)
      {
        x=r+rand()%(x2);
        y=r+rand()%(y2);
        if(network[x][y]==0)
         {
          network[x][y]=nodes_T+1;
          x=r+rand()%(x2);
          y=r+rand()%(y2);
          k=0;
         }
      }
        cout<<"pls";
      break;

case 4:
      x1=(max_X)*(pow(1-max_X,r));
      x2=int(x1);
      cout<<x2;
      y1=(max_Y)*(pow(1-max_Y,r));
      y2=int(y1);
     cout<<y2;
     for(i=0;k<(nodes_T);i++)
        {
         x=r+rand()%(x2);
         y=r+rand()%(y2);
         if(network[x][y]==0)
          {
            network[x][y]=k+1;
            k++;
          }
        }
        k=1;
      while(k)
      {
        x=r+rand()%(x2);
        y=r+rand()%(y2);
        if(network[x][y]==0)
         {
          network[x][y]=nodes_T+1;
          x=r+rand()%(x2);
          y=r+rand()%(y2);
```

```
            k=0;
           }
      }
          cout<<"pls";
      break;

      default:
      cout<<"njnhkjk";
      }
l=0;
clrscr();
for(i=0;i<max_X;i++)
  {
   cout<<"\n\n";
      for(j=0;j<max_Y;j++)
      {
       cout<<"    ";
       if(network[i][j]!=0&&network[i][j]!=nodes_T+1)
            {
         cout<<network[i][j];
            }
            if(network[i][j]==nodes_T+1)
            {
              cout<<"["<<nodes_T+1<<"-"<<nodes_T+nodes_M<<"]";
            }
   }
  }
 }
void show_cnv()
 {
  int i,j;
  cout<<"\nCommon Neighbor\n";
  for(i=0;i<length(neighbor[v]);i++)
   {
     cout<<"\n";
     for(j=0;j<length(cnv[i]);j++)
     cout<<" "<<cnv[i][j];
   }
 }


void main()
{
  time_t t;
```

```
 srand((unsigned) time(&t));
 clrscr();
 clock_t start;
 clock_t end;
 create_network();
 calculate_neighbor();
 cout<<"\n\nEnter the victim node\n\n";
 cin>>v;
 cv=.7*(length(neighbor[v]));
 cnb_IV();
 show_cnv();
 calculate_ap();
 start=clock();
 sybil_nodes();
 end=clock();
 float n,time;
 n=end-start;
 time=n/(CLK_TCK);
 cout<<"\nThe time calculated is "<<time;
 getch();
}
```

5.2.4 Screenshot C++ result

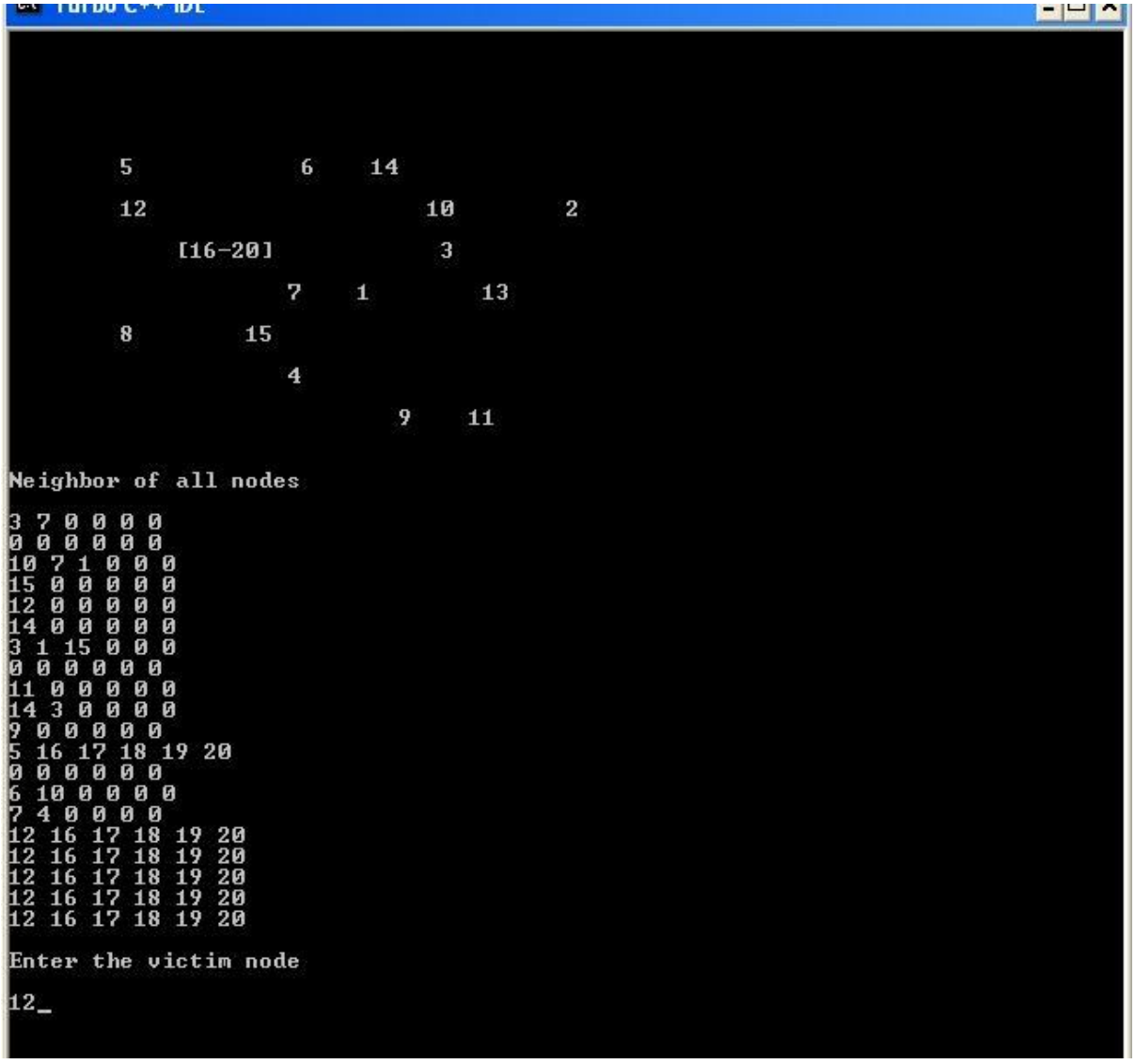


Figure 20: Nodes generated by random function

Figure 21: Detection of Sybil nodes

CONCLUSION

In an advance world of technologies, communications has reach further and are still growing. Even networking has made a huge turn. A key issue in a networking topology is ones security and Sybil attack is one of the security issues to be address. Our mechanism, detection by neighboring information have some limitation like it can't be accurate if the density of a node is sparse and if the Sybil node forges it neighbors information.

But detection by this mechanism is much more better compare to other approaches as in an ad hoc network where there is no centralized authority to monitor every node and this I have stated in the beginning. Also node being mobile in nature increases the complexity as node can disappear anytime from the communication range and even from the networking topology.

Still more is to be done in the field of networking and concerns related to securities.

Appendix A

Installing NS2 in LINUX platform
First, we download the ns-2 all-in-one file [54.4 MB].

$ wget http://nchc.dl.sourceforge.net/sourceforge/nsnam/ns-allinone-2.34.tar.gz
$ tar -xzvf ns-allinone-2.34.tar.gz
$ cd ns-allinone-2.34
$ sudo apt-get install build-essential autoconf automake libxmu-dev
$ ./install
If your ubuntu version is 9.10, you must change the variable of environment CC in makefile.in in otc2.11

$ export CC=gcc-4.3
Set environment variables
$ gedit ~/.bashrc

Add the following lines to the end of the file. Remember replace "/your/path" by the folder where you have stored extracted the ns-2 file (For example, if your Linux account name is purple, and you have extracted the file to your home directory, you have to change /your path to /home/purple)

# LD_LIBRARY_PATH
OTCL_LIB=/your/path/ns-allinone-2.34/otcl-1.13
NS2_LIB=/your/path/ns-allinone-2.34/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$USR_LOCAL_LIB

# TCL_LIBRARY
TCL_LIB=/your/path/ns-allinone-2.34/tcl8.4.18/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB

# PATH
XGRAPH=/your/path/ns-allinone-2.34/bin:/your/path/ns-allinone-2.34/tcl8.4.18/unix:/your/path/ns-allinone-2.34/tk8.4.18/unix
NS=/your/path/ns-allinone-2.34/ns-2.34/
NAM=/your/path/ns-allinone-2.34/nam-1.14/

PATH=$PATH:$XGRAPH:$NS:$NAM
Ensure that it immediately takes effect:

$ source ~/.bashrc
Note: the previous step is important; else you cannot successfully run ns-2.

(Alternatively, you may have to restart your X-windows, that is logout, and then login, or restart your computer.)

Now, the installation has been completed. Try:

$ ns
The "%" symbol appears on the screen. Type "exit" to quit.

Validation
To run the ns validation suite:

$ cd ns-2.34
$ ./validate

BIBLIOGRAPHY

1. Teerawat Issariyakul and Ekram Hossain, *Introduction to Network Simulator NS*, Springer, 2009.
2. Kuo-Feng Ssu, Wei-Tong Wang, Wen-Chung Chang, *Detecting Sybil Attacks in WSNs using neighboring information.*
3. J.R. Docucers, *The Sybil Attack*, in: Proceedings of the International Workshop on Peer-to-Pear Systems, March 2002.
4. C. Karlof & D. Wagner, *Secure routing in WSNs*: attacks and countermeasures, in: Proceedings of the IEEE International Workshop on Sensor Network Protocols and Applications, May 2003.
5. J. Newsome, E. Shi, D. Song, A. Perrig, T*he Sybil attack in sensor networks: analysis and defenses*, in: Proceeding of the international Symposium of Information Processing in Sensor Networks, April 2004.
6. M. Demirbas & Y. Song, *An RSSI-based scheme for Sybil attack detection in WSNs*, in: Proceedings of International Symposium on a World of Wireless, Mobile and Multimedia Networks, June 2006.
7. D. J. Huang, W.C. Teng, C.Y. Wang, H.Y. Huang, J.M. Hellerstein, *Clock skew based node identification in WSNs*, in: Proceedings of the IEEE Global Telecommunication Conference, November 2008.
8. The ns Manual, The VINT Project, http://www.isi.edu/nsnam/ns/tutorial/index.html
9. James F. Kurose and Keith W. Ross. *Computer Networking: A top Down Approach featuring Internet*, ADDISON-WESLEY, Sept 1999
10. Andrew S. Tanenbaum, *Computer Networks*, Fourth Edition, Prentice Hall, March 17, 2003
11. Altman and Jimene, *NS for Beginners*, Lecture Notes 2003
12. Michael Welzl, *Tutorial: The ns-2 Network Simulator*
13. Jianping Wang, *ns-2 Tutorial (2)*, Lecture Notes 2004
14. Jerry Banks, John Carson, Barry Nelson, David Nicol, P. Shahabudeen, *Discrete-Event System Simulation*, Fourth Edition, Prentice Hall, 2009
15. http://nile.wpi.edu/NS/
16. P. Wu, “ns Bench-Graphical User Interface for Network Simulator”, http://www.mnlab.csdepal.edu/projects/nsbench/, 2005