

**Novel Encryption and Lossless Compression Technique for Secure
Data Transmission using NS2**

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering degree in Electronics and Communication Engineering

By
DAYAKA RAJA (Reg. No. 37130104)
D SAI RAGHU RAMI REDDY (Reg. No. 37130117)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS**

**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade “A” by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

APRIL – 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)



Accredited with “A” grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Dayaka Raja (REG.NO 37130104)** and **D Sai Raghu Rami Reddy (REG.NO 37130117)** who carried out the project entitled “ **Novel Encryption and Lossless Compression Technique for Secure Data Transmission using NS2**” under my supervision from

Internal Guide
Ms. YOGALAKSHMI (Asst PROFF),,

Head of the Department
Dr . T. RAVI, M.E., Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

We **DAYAKA RAJA (Reg. No.37130104)** and **D SAI RAGHU RAMI REDDY (Reg. No.37130117)** hereby declare that the Project Report entitled “**Novel Encryption and Lossless Compression Technique for Secure Data Transmission Using NS2**” done by us under the guidance of **Ms. YOGALAKSHMI**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in **ELECTRONICS AND COMMUNICATION ENGINEERING**.

DATE:
PLACE:

SIGNATURE OF THE CANDIDATE

- 1.**
- 2.**

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N.M. NANDITHA, M.E., Ph.D.**, Dean, School of Electrical and Electronics and **Dr. T. RAVI, M.E., Ph.D.**, Head of the Department, Dept. of Electronics and Communication Engineering for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Ms. YOGALAKSHMI (Asst PROFF)**., for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

We wish to express our thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Communication Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

This paper examines various issues of Wireless Sensor Network (WSN) and the pertinence of the Advanced Encryption Standard Security in WSN is a more noteworthy test in WSN because of the preparing impediments of sensor hubs and nature of remote connections. Broad utilization of WSNs is offering ascend to various sorts of dangers. To guard against the dangers appropriate security plans are required. Customarily security is carried out through equipment or programming and is by and large accomplished through cryptographic strategies. Restricted zone, nature of connections, restricted handling, force and memory of WSNs prompts exacting limitations on the determination of cryptographic methods. In this paper we examine the got data transmission over the IOT organization and looking at the two changed cryptography calculations RC4 and Advanced Encryption Standard (AES) are the best applicant because of its more modest key size. High security notwithstanding of more modest key size brings about region and force productive crypto frameworks.

Table of Contents

1. INTRODUCTION.....	1
1.1 : METHOD OF APPROACHING:	2
2. LITERATURE SURVEY	4
3. AIM AND SCOPE	6
3.1 : EXISTING SYSTEM.....	6
3.2 : PROPOSED SYSTEM	7
3.3: NEIGHBOR DISCOVERY BLOCK.....	8
3.4 BROADCASTING NODES.....	9
4. ALGORITHMS AND IMPLEMENTATION	10
4.1: ALGORITHMS	10
4.1.1 AES Algorithm (Advanced Encryption Standard)	10
4.1.2 RC4 Algorithm (Rivest Cipher 4).....	13
4.2 : Implementation.....	16
4.2.1 OTCL Essentials.....	16
4.2.2 Assigning Esteems to Factors	17
4.2.3 procedures.....	17
4.2.4 Files and records	19
4.2.5 Calling Subprocesses	19
4.2.6 Creating the Geography.....	20
4.2.7 Nodes	21
4.2.8 Agents, applications and traffic sources.....	21
4.2.9 Traffic Sinks	24
4.2.10 Links	25
4.2.11 Tracing and checking.....	26
4.2.12 Traces.....	27
4.2.13 Monitors.....	29
4.2.14 Controlling the recreation.....	30
4.3 : Simple Stimulation Example	32
4.4 Extending NS: Creating another Agent	39

4.5 The header record.....	39
4.6 Necessary changes in NS	48
5. RESULTS AND DISCUSSION	49
5.1 Edit packet.h file.....	49
5.2 Edit ns-packet.tcl document	50
5.3 Edit ns-default.tcl.....	51
5.4 Edit Makefile.in.....	51
5.5 Recompile NS	52
5.6 Simple example using MyPing agent.....	52
5.7 AES – NAM Results.....	53
5.8 AES - Parameter Analysis	56
5.9 RC4 - NAM Results	58
5.10 RC4 - Parameter Analysis.....	60
6. SUMMARY AND CONCLUSION.....	64

LIST OF FIGURES

Figure 4.1 Bit State and Cipher key in Columns	18
Figure 4.2 Transformation Rounds in AES	18
Figure 4.3 S-box Transformation	19
Figure 4.4 Shift row Transformation	19
Figure 4.5 Figure link in NS2	31
Figure 4.6 Figure Link in NS2 when tracing is enabled	33
Figure 4.7 A simple Network topology and simulation Scenario	39
Figure 4.8 Figure of NS directory structure.....	55
Figure 5.1 Network Representation NAM output file(AES)	62
Figure 5.2 Authentication between Nodes.....	62
Figure 5.3 Routing in Network	63
Figure 5.4 Receiving data at receiver node from transmitter node	64
Figure 5.9 (a) Packet delivery ratio.....	65
Figure 5.9 (b) Throughput.....	66
Figure 5.9 (c) Packet Loss Ratio	66
Figure 5.9 (d) Comparision of Energy.....	67
Figure 5.9 (e) Response Time.....	67
Figure 5.9.1 Network Representation NAM Output(RC4)	68
Figure 5.9.2 Authentication between Transmitter and Receiver nodes.....	69
Figure 5.9.3 Routing path between Source and Destination.....	69
Figure 5.10(a) Throughput	70
Figure 5.10 (b) Packet Loss Ratio	70
Figure 5.10 (c) Packet Delivery Ratio.....	71
Figure 5.10(d) Comparision of Energy.....	71
Figure 5.10(e) Response Time.....	72

1.INTRODUCTION

Web of Things permits us to feel that we will actually want to interface with each sort of gadget in our environmental factors utilizing the Internet. This will permit us to impart or interface with any sort of gadget around us from anyplace on the planet. The eventual fate of IoT is really subject to a couple of the renowned and key advances of today like Wireless Sensor Networks, Miniaturization, and Nanotechnology. Remote Sensor Networks are continually going to be the significant part in such an IoT applications. In this time, Wireless Sensor Networks is the innovation whose development is expanding definitely and is viewed as the most appealing and developing field of exploration nowadays. They are a piece of numerous enormous applications these days, similar to crisis reaction, the board, and reconnaissance of combat zone, investigation, and observing of the climate data, checking in clinics, and stock administration. Remote sensor networks are the main decision these days due to their unwavering quality, easy method of organization, less expense, and capacity to investigate those spots which far from individuals. Giving security to a Wireless Network is an exceptionally troublesome assignment. Furthermore, then again, the principle issue is, it is a lot of helpless against assaults. There are various assaults conceivable on Wireless Sensor Networks. Confusion Attacks are the most well known sort of Denial-of administration assaults conceivable on Wireless Sensor Networks. This assault should be possible in various manners. A noxious hub can upset a solid course of parcels from source to objective, making the bundles from a specific hub not arrive at the objective. Subsequently, it causes an expansion in the deferral of the parcels to arrive at the objective or it is conceivable they may not arrive at the objective by any means. This further debases the throughput of the organization. Here, in this paper, we have introduced a novel procedure to recognize and forestall the confusion assault. In the outcomes, it is shown that how the throughput and deferral of the organization are diminishing when a malignant hub is available in the organization and how there is an impressive reduction in the energy utilization of the organization after the location and disengagement of the low-execution hub from the organization.

1.1 : METHOD OF APPROACHING:

A new energy and security aware route selection (ESARS) protocol for WSNs are proposed. The first part of ESARS protocol is responsible for providing an energy-efficient route selection of a path among a set of available multiple paths with the objective of avoiding early sensors' death and extending the NL. On the basis of the review of the literature, we

conclude that using the residual energy is essential in designing energy-efficient route selection protocols as it measures the remaining battery power in the sensors, which is very critical to decide if a node is close to die or not. Another important issue is to avoid routes with nodes that has energy level close to critical and may die soon. Depending only on the average residual energy (of the nodes on a given path) will not address the issue of close to die nodes. Hence, these paths have to be treated specifically by examining the residual energy at each node.

A vital perspective, which was not especially tended to in the current energy-efficient course determination conventions, is the basic hub circumstance, whose disappointments may segment the arrange or make correspondence bottlenecks. Researchers showed that the critical nodes are most likely to be the nodes with the highest node degree, which is defined as the number of directly connected neighbors of a node. Hence, nodes with high node degree are likely to participate in many paths in the network. A node with moderate residual energy is not as critical as a node with the same residual energy but higher node degree. Accordingly, the first part of the proposed protocol considers the node degree with the residual energy to form an original metric for route selection.

The second part of ESARS protocol is responsible for picking the optimum security level (SL) for the selected path from part one. As per Stavrou and Pitsillides, the primary security necessities in WSNs multipath steering are accessibility, dependability, confirmation, uprightness, and newness. In ESARS convention, security is characterized into a few levels with the end goal that the higher SL utilizes more grounded cryptography (encryption and validation) and consequently burns-through more energy. Additionally, the survey of the writing on multipath-based security conventions in WSNs affirms that trust assumes a significant part in course choice. Sensors with low trust level may cause damage to the WSN by dropping, delaying, modifying, or masquerading packets. Hence, routes with untrusted sensor are excluded by ESARS. Additionally, based on the attributes of the chose way and the hubs on that way, ESARS convention utilizes a novel standards to pick the ideal SL. The thought is to apply a more grounded SL with low normal trust level courses. In addition, if the selected route has low average residual energy, then ESARS protocol will not overload the nodes with heavy cryptography. Moreover, longer routes are more susceptible to security attacks such as falsification of data and message modification; hence, ESARS protocol

assigns stronger authentication for comparatively long selected route. However, another fascinating element of ESARS convention is that it utilizes edge boundaries which could be changed per the requirements of the application wherein the sensors are utilized. The proposed convention is altogether assessed and contrasted and other course choice conventions utilizing broad recreations and scientific investigation.

The experimental results and analysis show that ESARS protocol minimizes the sensors death rate and provides the longest NL. It also has the best packet delivery ratio and the highest SL.

2.LITERATURE SURVEY

1. Akyildiz I, Su W, Sankarasubramaniam Y, Cayirci E. Wireless sensor networks: a survey. *Computer Networks* 2002; 38(4): 393- 422.
a. Crossref Web of Science@Google Scholar
2. Dietrich I, Dressler F. On the lifetime of wireless sensor networks. *ACM Transactions on Sensor Networks* 2009; 5(1): 5:1- 5:39.
a. Crossref Web of Science@Google Scholar
3. Sha K, Gehlot J, Greve R. Multipath routing techniques in wireless sensor networks: a survey. *Wireless Personal Communications* 2013; 70(2): 807- 829.
Crossref Web of Science@Google Scholar
4. Tarique M, Tepe KE, Adibi S, Erfani S. Survey of multipath routing protocols for mobile ad hoc networks. *Journal of Network and Computer Applications* 2009; 32(6): 1125- 1143.
5. Vergados DJ, Pantazis NA, Vergados DD. Energy- efficient route selection strategies for wireless sensor networks. *Mobile Networks and Applications* 2008; 13(3- 4): 285- 296.
Web of Science@Google Scholar
6. Boukerche A, Turgut B, Aydin N, Ahmad MZ, Blni L, Turgut D. Routing protocols in ad hoc networks: a survey. *Computer Networks* 2011; 55(13): 3032- 3080.
Crossref Web of Science@Google Scholar
7. Wang G, Chen S, Lu H, Lin M. Pa- gfdp: an algorithm enhancing reliability of wsns. *WSEAS Transactions on Computers* 2012; 11(12): 445- 454.
Google Scholar
8. Kim TH, Tipper D, Krishnamurthy P. Connectivity and critical point behavior in mobile ad hoc and sensor networks, *Proceedings of IEEE ISCC, Sousse, 2009*; 153- 158.
Google Scholar
9. Stavrou E, Pitsillides A. A survey on secure multipath routing protocols in wsns. *Computer Networks* 2010; 54(13): 2215- 2238.
Crossref Web of Science@Google Scholar
10. Wang C, Cai S, Li R. Aodvsec: a multipath routing protocol in ad- hoc networks for improving security, *Proceedings of IEEE MINES, Hubei, 2009*; 401- 404.
Google Scholar
11. Tang, L.; Lu, Z.; Cai, J.; Yan, J. An Equilibrium Strategy-Based Routing Optimization Algorithm for Wireless Sensor Networks. *Sensors* 2018, 18, 3477.

12. Hajji, F.E.; Leghris, C.; Douzi, K. Adaptive Routing Protocol for Lifetime Maximization in Multi-Constraint Wireless Sensor Networks. *J. Commun. Inf. Netw.* 2018, 3, 67-83.
13. Wang, X.; Li, D.; Zhang, X.; Cao, Y. MCDM-ECP: Multi Criteria Decision Making Method for Emergency Communication Protocol in Disaster Area Wireless Network. *Appl. Sci.* 2018, 8, 1165.
14. Khan, B.M.; Bilal, R.; Young, R. Fuzzy-TOPSIS Based Cluster Head Selection in Mobile Wireless Sensor Networks. *J. Electr. Syst. Inf. Technol.* 2017, 5, 928-943.
15. Jiang, H.; Sun, Y.; Sun, R.; Xu, H. Fuzzy-Logic-Based Energy Optimized Routing for Wireless Sensor Networks. *Int. J. Distrib. Sens. Netw. IJDSN* 2013, 9, 264-273.

3. AIM AND SCOPE

- The remote sensor network concentrated in this paper is basically utilized for occasion disclosure and data assortment. Because of the restricted correspondence distance between sensor hubs, the hubs for the most part communicate information to the base station through multi-jumps.
- Instructions to discover the way with the least energy utilization from a source hub to a sink hub is a need for directing calculations. The geography of WSN is appeared in Figure 1. The data assortment terminals are viewed as basic hubs, which send the gathered information to a sink hub through multi-bounce sending.
- Assume in the sensor network with a territory of $L \times L$, there are various normal hubs and a solitary sink hub situated in the middle, and all hubs are not, at this point moved after sending.
- The area between hubs is resolved . All sensor hubs are isomorphic, with directing, sending and accepting capacities, any two hubs can impart in single-jump or multi-bounce, and the underlying state is something very similar.
- The underlying heap of the hubs is 0, and the underlying energy is E_0 . The data trade strategy between hubs is as . That is, every hub has an interesting identifier (ID), which keeps a support to store data like remaining energy, parcel ID, next-bounce ID, sender IDs and so forth This data is refreshed continuously as the forward neighbor changes.
- It is noticed that the traffic line length of the hubs is restricted, and the information parcel is prepared in first-in first-out (FIFO) mode.

1.1 : EXISTING SYSTEM:

The improvement of proficient remote correspondence and the advancement of electronic data innovation, the remote sensor organization (WSN) is generally utilized in different fields in light of its minimal effort, scaling down and multi-work qualities.

Notwithstanding, much of the time, the hubs in WSN are controlled by batteries and are generally sent in automated open air or more hazardous conditions, which make it awkward to recharge energy. The expense of repetitive organization and hub substitution is additionally generally high. Accordingly, a productive directing system is expected to limit network energy utilization and draw out the organization lifetime. Since the energy of the sensor hub is principally utilized for information gathering and transmission, the conventional steering methodology primarily thinks about how to use the most brief way to move information from the source hub to the objective as fast as could be expected. Be that as it may, in the energy-compelled sensor organization, a lot of information is sent from the source hub to sink by "many-to-one" mode, which effectively aims genuine "channel impact" and "energy opening" issues.

2.2 : PROPOSED SYSTEM:

The enormous interest for asset restricted Wireless Sensor Networks is the requirement for reliable and viable security gadgets for them has improved different however its application is a non-paltry errand. Limitations in handling speed, battery force, transmission capacity, and capacity oblige the relevance of present cryptography methods for WSNs. In this paper, the interaction is proposed AES-RC4 broke down and their appropriateness for asset restricted remote organization security are related based on execution standards, for example, normal energy of each hub and execution time. Utilizing recreation tests and coherent models, we give a huge examination and thoughts on the common sense possibility of these cryptographic cycles in sensor organizations to assist creators with foreseeing security execution under a bunch of constraints for remote sensor organizations.

3.3: NEIGHBOR DISCOVERY BLOCK

On network bootstrapping or with the presentation of each new hub, the Neighbor Discovery Block is enacted. In this stage, each hub signs and broadcasts a one-bounce hi message to acquaint itself with its adjoining hubs. The welcome message have the accompanying configuration—{Message Type = 1 [one byte], Sender ID [one byte], MAC [x bytes]} of the bundle, if appropriate, by checking its MAC utilizing the sender's public

key (node ID) as the confirmation key for PBC, or utilizing the symmetric key for ECC, as set by the director in the Initialization Block. In the event that the bundle is valid, the beneficiary hub adds the sender in its neighbors rundown and answers utilizing a marked "unicast" hi answer parcel back to the sender to affirm the neighborhood between them. The welcome answer message has a similar organization as the welcome message with the Message Type = 2.

Additionally, every fixed time, set and synchronized by the BS, all hubs broadcasts to its neighbors hello_keep_alive message. The time frame is set by the head in the Initialization Block, and it is picked to be various of times the time of sending the action reports. This welcome keep-alive message has precisely the same arrangement as a welcome message with the Message Type = 3, and it is utilized to ensure that a hub actually exists to keep the BS refreshed with right data. Endless supply of a welcome keep-alive message, the accepting hub simply invigorates the situation with its neighbors, without answering. Note that in the event of a verification conspire set, a wide range of hi messages should be marked and checked for confided in neighbors' personalities.

4.4 BROADCASTING NODES

Another kind of assault is a preliminary to upset the organization by continually communicating parcels to every one of its neighbors. This sort is recognized and detached in two phases. In the first place, the impact of the transmission is privately eliminated straightforwardly since the entirety of its non DL neighbors drop this parcel. Concerning the high transmission pace of the hub, this is identified by the BS and chooses to put the hub waiting on the post trial process or straightforwardly separate it relying upon the overhead of the time frame from what it ought to be.

4.ALGORITHMS AND IMPLEMENTATION

4.1 : ALGORITHMS

1.1.1 AES Algorithm (Advanced Encryption Standard)

The AES (Advanced Encryption Standard) is a symmetric square code that can encipher or translate the data. The AES calculation utilizes three distinctive key lengths 128, 192, and 256 pieces to scramble or unscramble information in squares of 128 pieces. The AES calculation was created by Joan-Daemen Proton World International and Vincent Rijmen Katholieke University at Leuven, Belgium. The key length will be reached out in products of 32 cycle. AES is an iterated block figure, where both key the Initial data sources go through various rounds of change prior to creating the first code message. The consequences of the middle of the road figure are the state. The code key and the square are introduced by a variety of sections, where each exhibit has four lines and a segment addresses a solitary byte. A cluster addressing the state will have a Nb section, where the estimations of Nb are 4, 6, and 8 corresponding to 128, 192, and 256-bit blocks. The exhibit addressing the code key will have Nk sections, where the Nk estimations of 4, 6, and 8 will address the relating key length. An illustration of 128-cycle state (Nb=4) and 192-piece figure key (Nk=6) are appeared in Fig1

s _{0,0}	s _{0,1}	s _{0,2}	s _{0,3}	k _{0,0}	k _{0,1}	k _{0,2}	k _{0,3}	k _{0,4}	k _{0,5}
s _{1,0}	s _{1,1}	s _{1,2}	s _{1,3}	k _{1,0}	k _{1,1}	k _{1,2}	k _{1,3}	k _{1,4}	k _{1,5}
s _{2,0}	s _{2,1}	s _{2,2}	s _{2,3}	k _{2,0}	k _{2,1}	k _{2,2}	k _{2,3}	k _{2,4}	k _{2,5}
s _{3,0}	s _{3,1}	s _{3,2}	s _{3,3}	k _{3,0}	k _{3,1}	k _{3,2}	k _{3,3}	k _{3,4}	k _{3,5}

Fig 4.1 Bit state and Cipher key in Columns

The number of transformation rounds (Nr) is a function of block length and key length and is shown in the Fig2 below

No. of Rounds Nr		Block Size		
		128 bits Nb = 4	192 bits Nb = 6	256 bits Nb = 8
Key Size	128 bits Nk = 4	10	12	14
	192 bits Nk = 6	12	12	14
	256 bits Nk = 8	14	14	14

Fig 4.2 Transformation rounds in AES

The AES calculation comprises of four operational stages, they are a) Add-round key change b) Substitute byte stage c) Shift lines d) Mix section change. The exhibits S and S' address the state when the change.

a) The Substitute byte change: The substitute byte change works on every one of the state bytes autonomously and changes the worth. The RIJNDAEL S-box is utilized for the byte replacement change stage. The S-enclose particulars are accessible the AES determination. For instance, the information state byte estimation of EA will be subbed by another estimation of 87, etc. Fig3 shows the substitute byte change

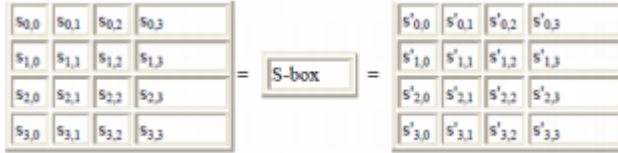


Fig 4.3 S-box transformation

b) Shift column change It is a straightforward stage on the substitute by changed lattice. The principal line of the state isn't changed. For the subsequent column, a 1-byte roundabout left move is performed. In the third column, a 2-byte round left move is performed, etc. Fig4 addresses the forward move line change.

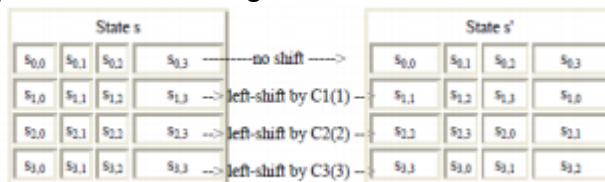


Fig 4.4 Shift row transformation

c) Mix Column change The forward blend section change works on every segment separately. Every byte of a section is planned into another worth that is a component of the relative multitude of four bytes in that segment. The blend segment activity utilizes a predefined polynomial following up on the four qualities all at once in the Galois field GF (28). Here the expansion is addressed by the bitwise XOR activity and duplication is by 1-cycle left move activity followed by the restrictive bitwise XOR activity.

d) Add Round Key Transformation In the forward add round key change, called the Add Round Key, the 128 bits of the state or bitwise XORed with the 128 bits of the round key. We need to infer an alternate key for each round which is to be applied during each round of the encryption activity. These keys are known as the round keys and each will be a similar length of the square. Nb 32bit words are indicated by W. The AES unique particulars characterize a vital timetable by which the first code key is utilized to shape an extended key. The extended key size is equivalent to the square size times the quantity of encryption adjusts augmented by an estimation of 1, which will give $Nr+1$ various keys.

For instance a 128 bit AES key will be extended to a 44 32bit word key that can be utilized for 10 rounds of change. The first code key involves the main part of the extended key and is utilized to create the excess key material. The AES 192 piece key will 52 32 cycle - word key and the 256 digit variant will have 60 32 bit word key that can be utilized for the 14 rounds of change. Fig 4 addresses the extended key rendition utilized in the AES 128 digit key length. Review that the round key is the length of the square utilized. The key development calculation is utilized to infer the extended key that is utilized as a critical material for each round of change.

2.1.2 RC4 Algorithm (Rivest Cipher 4)

RC4 is a stream cipher type. It estimates unit or data at one time. Unit or data is a byte or even at times bits. Along these lines, the encryption or decoding can be executed on the length of the variable. This calculation doesn't need to stand by a specific measure of information contribution before it is prepared or add additional bytes to scramble. The model is RC4. Another sort is a square code that measures simultaneously a specific measure of information (ordinarily a 64-cycle or 128-bit blocks) For instance, Blowfish, DES, Gost, Idea, RC5, Safer, Square, Twofish, RC6, Loki97, and so on

RC4 is a restrictive symmetric encryption stream made by RSA Data Security, Inc. The appropriation is started from a source code that is accepted to be as RC4 and distributed secretly in 1994 . The distributed calculation is inseparable from the execution of the RC4 on the authority item. RC4 is broadly utilized in numerous applications and is usually communicated exceptionally protected. RC4 key age is separated into a few phases. depicts the phases of the RC4.S-Box instatement is to orchestrate the secret word involved to be the byte exhibit. In the interim, the change is to do the new byte cluster to as long as the plaintext accessible. The new key will be scrambled to the plaintext. It produces the ciphertext.

In RC4 cryptography, this calculation has a S-Box, S [0], S [1], ..., S [255], which contains a change of the numbers 0 to 255 where the stage is a capacity key K, with a compelling length. Figure 3describes the age of the S-Box

The organization of the S-Box on a similar RC4 may happen. This game plan brought about calculations are defense less. It happens on the grounds that the estimation of a similar pseudorandom frequently raised over and over, this happens on the grounds that the client key is rehashed to fill 256 bytes cluster . Albeit this strategy permits the utilization of variable length can arrive at 256 bytes, there nobody utilize such length. On the off chance that the client possesses eight bytes key length, it will be rehashed multiple times to fill the byte exhibit.

RC4 encryption is the XOR interaction between information bytes and pseudorandom byte stream created from the key ; at that point the assailant will be feasible to decide a portion of the first messages by performing byte XOR measure on the two arrangements of code bytes when some obscure plaintext byte. For instance, "A" effectively blocked two diverse message scrambled utilizing a stream figure calculation utilizing a similar key. "A" performs XOR to the ciphertext cycle is effectively taken to wipe out the impact of key arrangement. In the event that the "A" figured out how to discover the plaintext of one of the scrambled message is then "A" will handily get another message plaintext without knowing the key.

4.2 : Implementation

NS (version 2) is an open-source network reenactment instrument. It is an item-arranged, discrete occasion-driven test system written in C++ and Otcl. The essential utilization of NS is in-network investigations to reenact different kinds of wired/remote neighborhood and wide territory organizations; to actualize network conventions like TCP and UDP, traffic source conduct like FTP, Telnet, Web, CBR, and VBR, switch line the board system like Drop Tail, RED, and CQB, steering calculations like Dijkstra, and some more.

Ns2 is written in C++ and Otcl to isolate the control and information way usage. The test system underpins a class chain of command in C++ (the accumulated order) and a comparing progression inside the Otcl translator (deciphered progression).

The motivation behind why ns2 utilizes two dialects is that various assignments have various prerequisites: For instance reenactment of conventions requires productive control of bytes and bundle headers making the run-time speed vital. Then again, in-network contemplates where the point is to differ a few boundaries and to rapidly inspect various situations an opportunity to change the model and run it again is more significant.

In ns2, C++ is utilized for definite convention usage and when all is said in done for such situations where each parcel of a stream must be prepared. For example, in the event that you need to execute another lining discipline, C++ is the language of decision. Once, then again, is reasonable for design and arrangement. Otcl runs gradually, yet it tends to be changed rapidly making the development of reproductions simpler. In ns2, the aggregated C++ items can be made accessible to the Otcl mediator. Thusly, the instant C++ items can be controlled from the Tcl level.

1.2.1 OTCL Essentials

This part presents the sentence structure and some fundamental orders of the Otcl language that are utilized by ns2. It is essential to see how Otcl functions prior to moving to the part that manages the formation of the real reenactment situation.

2.2.2 Assigning Esteems to Factors

In tcl, qualities can be put away to factors and these qualities can be additionally utilized in orders:

```
set a 5
```

```
set b [expr $a/5]
```

In the primary line, the variable `a` is appointed the worth "5". In the subsequent line, the aftereffect of the order `[expr $a/5]`, which rises to 1, is then utilized as a contention to another order, which thus appoints an incentive to the variable `b`. The "\$" sign is utilized to acquire a worth contained in a variable and square sections are a sign of an order replacement.

3.2.3 procedures

One can characterize new systems with the `proc` order. The primary contention to `proc` is the name of the strategy and the subsequent contention contains the rundown of the contention names to that methodology. For example a methodology that computes the amount of two numbers can be characterized as follows:

```
proc total {a b} {
```

```
    expr $a + $b
```

```
}
```

The following system figures the factorial of a number:

```
proc factorial a {
```

```
    on the off chance that {$a <= 1} {
```

bring 1 back

}

#here the system is called once more

expr \$x * [factorial [expr \$x-1]]

}

It is likewise conceivable to give an unfilled string as a contention list. Notwithstanding, for this situation the factors that are utilized by the strategy must be characterized as worldwide. For example:

proc whole {} {

worldwide a b

expr \$a + \$b

}

4.2.4 Files and records

In tcl, a document can be opened for perusing with the order:

```
set testfile [open test.dat r]
```

The primary line of the document can be put away to a rundown with an order:

```
gets $testfile list
```

Presently it is conceivable to acquire the components of the rundown with orders (numbering of components begins from 0) :

```
set first [lindex $list 0]
```

```
set second [lindex $list 1]
```

Essentially, a record can be composed with a puts order:

```
set testfile [open test.dat w]
```

```
puts $testfile "testi"
```

5.2.5 Calling Subprocesses

The order executive makes a subprocess and hangs tight for it to finish. The utilization of executive is like providing an order line to a shell program. For example, to eliminate a record:

```
executive rm $testfile
```

The executive order is especially valuable when one needs to call a tcl-content from inside another tclscript. For example, to run the tcl-content example.tcl on different occasions with the estimation of the boundary "test" going from 1 to 10, one can type the accompanying lines to another tcl-content:

```
for {set ind 1} {$ind <= 10} {incr ind} {
```

```
set test $ind
```

```
executive ns example.tcl test
```

```
}
```

6.2.6 Creating the Geography

To have the option to run a reproduction situation, an organization geography should initially be made. In ns2, the geography comprises of an assortment of hubs and connections.

Before the geography can be set up, another test system object should be made toward the start of the content with the order:

```
set ns [new Simulator]
```

The test system object has part works that empower making the hubs and the connections, interfacing specialists and so forth All these fundamental capacities can be found from the class Simulator. When utilizing capacities having a place with this class, the order starts with "\$ns", since ns was characterized to be a handle to the Simulator object.

7.2.7 Nodes

New hub articles can be made with the order:

```
set n0 [$ns node]
```

```
set n1 [$ns node]
```

```
set n2 [$ns node]
```

```
set n3 [$ns node]
```

The part capacity of the Simulator class, called "hub" makes four hubs and appoints them to the handles n0, n1, n2 and n3. These handles can later be utilized when alluding to the hubs. In the event that the hub isn't a switch however an end framework, traffic specialists (TCP, UDP and so on) and traffic sources (FTP,CBR and so forth) should be set up, i.e, sources should be connected to the specialists and the specialists to the hubs, individually.

8.2.8 Agents, applications and traffic sources

The most widely recognized specialists utilized in ns2 are UDP and TCP specialists. If there should be an occurrence of a TCP specialist, a few kinds are accessible. The most widely recognized specialist types are:

- Agent/TCP - a Tahoe TCP sender
- Agent/TCP/Reno - a Reno TCP sender
- Agent/TCP/Sack1 - TCP with specific affirmation

The most well-known applications and traffic sources given by ns2 are:

- Application/FTP - produces mass information that TCP will send
- Application/Traffic/CBR - produces bundles with a steady piece rate
- Application/Traffic/Exponential - during off-periods, no traffic is sent. During on-periods, bundles are produced with a consistent rate. The length of both on and off-periods is dramatically circulated.
- Application/Traffic/Trace - Traffic is produced from a follow record, where the sizes and interarrival seasons of the parcels are characterized.

Notwithstanding these instant applications, it is conceivable to produce traffic by utilizing the techniques given by the class Agent. For instance, on the off chance that one needs to send information over UDP, the technique

```
send(int nbytes)
```

can be utilized at the tcl-level given that the udp-specialist is first designed and appended to some hub.

The following is a finished illustration of how to make a CBR traffic source utilizing UDP as transport convention and connect it to hub n0:

```
set udp0 [new Agent/UDP]
```

```
$ns connect specialist $n0 $udp0
```

```
set cbr0 [new Application/Traffic/CBR]
```

```
$cbr0 append specialist $udp0
```

```
$cbr0 set packet_size_ 1000
```

```
$udp0 set packet_size_ 1000
```

```
$cbr0 set rate_ 1000000
```

A FTP application utilizing TCP as a vehicle convention can be made and appended to hub n1 similarly:

```
set tcp1 [new Agent/TCP]
```

```
$ns append specialist $n1 $tcp1
```

```
set ftp1 [new Application/FTP]
```

```
$ftp1 append specialist $tcp1
```

```
$tcp1 set packet_size_ 1000
```

The UDP and TCP classes are both kid classes of the class Agent. With the articulations [new Agent/TCP] and [new Agent/UDP] the properties of these classes can be consolidated to the new items udp0 and tcp1. These items are then joined to hubs n0 and n1. Then, the application is characterized and appended to the vehicle convention. At long last, the design boundaries of the traffic source are set. In the event of CBR, the traffic can be characterized by boundaries rate_ (or proportionally interval_, deciding the interarrival season of the parcels), packetSize_ and random_. With the random_ boundary it is conceivable to add some irregularity in the interarrival seasons of the bundles. The default esteem is 0, implying that no irregularity is added.

9.2.9 Traffic Sinks

In the event that the data streams are to be ended without handling, the udp and tcp

sources must be associated with traffic sinks. A TCP sink is characterized in the class Agent/TCPSink and a UDP sink is characterized in the class Agent/Null.

A UDP sink can be appended to n2 and associated with udp0 in the accompanying manner:

```
set invalid [new Agent/Null]
```

```
$ns join specialist $n2 $null
```

```
$ns associate $udp0 $null
```

A standard TCP sink that makes one affirmation for every a got parcel can be appended to n3 and associated with tcp1 with the orders:

```
set sink [new Agent/Sink]
```

```
$ns join specialist $n3 $sink
```

```
$ns associate $tcp1 $sink
```

There is likewise a more limited approach to characterize associations between a source and the objective with the order:

```
$ns make association <srctype> <src> <dsttype> <dst> <pktclass>
```

For instance, to make a standard TCP association somewhere in the range of n1 and n3 with a class ID of 1:

```
$ns make association TCP $n1 TCPSink $n3 1
```

One can undoubtedly make a few TCP-associations by utilizing this order inside a for-

circle.

10.2.10 Links

Connections are needed to finish the geography. In ns2, the yield line of a hub is executed as a component of the connection, so when making joins the client likewise needs to characterize the line type.

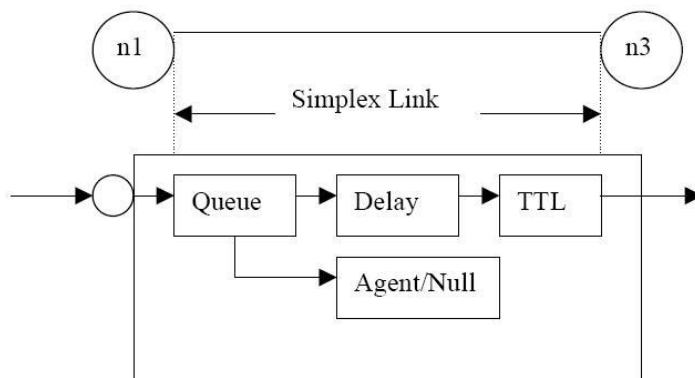


Fig 4.5 Figure Link in ns2

This Figure shows the development of a simplex connection in ns2. On the off chance that a duplex-interface is made, two simplex connections will be made, one for every course. In the connection, bundle is first enqueued at the line. After this, it is either dropped, passed to the Null Agent and liberated there, or dequeued and passed to the Delay object which mimics the connection delay. At long last, the TTL (time to live) esteem is determined and refreshed.

Connections can be made with the accompanying order:

```
$ns duplex/simplex-connect endpoint1 endpoint2 data transfer capacity defer line type
```

For instance, to make a duplex-connect with DropTail line the executives somewhere in the range of n0 and n2:

```
$ns duplex-interface $n0 $n2 15Mb 10ms DropTail
```

Making a simplex-interface with RED line the executives somewhere in the range of n1 and n3:

\$ns simplex-interface \$n1 \$n3 10Mb 5ms RED

The qualities for data transmission can be given as an unadulterated number or by utilizing qualifiers k (kilo), M (super), b (cycle) and B (byte). The postponement can likewise be communicated in a similar way, by utilizing m (milli) and u (mikro) as qualifiers. There are a few line the board calculations actualized in ns2, yet in this activity just DropTail and RED will be required.

11.2.11 Tracing and checking

To have the option to compute the outcomes from the reenactments, the information must be gathered by one way or another. NS2 upholds two essential checking abilities: follows and screens. The follows empower recording of parcels at whatever point an occasion, for example, bundle drop or appearance happens in a line or a connection. The screens give a way to gathering amounts, for example, number of parcel drops or number of showed up bundles in the line. The screen can be utilized to gather these amounts for all bundles or only for a predetermined stream (a stream screen).

12.2.12 Traces

All occasions from the reenactment can be recorded to a document with the accompanying orders:

```
set trace_all [open all.dat w]
```

```
$ns follow all $trace_all
```

```
$ns flush-follow
```

```
close $trace_all
```

To start with, the yield document is opened and a handle is connected to it. At that point the occasions are recorded to the document determined by the handle. At long last, toward the finish of the recreation the follow support must be flushed and the record must be shut. This is typically finished with a different completion system. On the off chance that connections are made after these orders, extra articles for following (EnqT, DeqT, DrpT and RecvT) will be embedded into them.

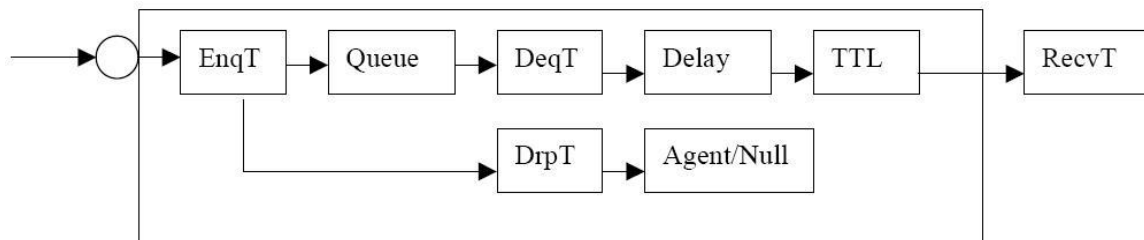


Fig 4.6 Figure Link in ns2 when tracing is enabled

ese new items will at that point keep in touch with a follow record at whatever point they get a bundle. The configuration of the follow document is following:

```
+ 1.84375 0 2 cbr 210 - 0 0.0 3.1 225 610
```

```
- 1.84375 0 2 cbr 210 - 0 0.0 3.1 225 610
```

```
r 1.84471 2 1 cbr 210 - 1 3.0 1.0 195 600
```

r 1.84566 2 0 ack 40 - 2 3.2 0.1 82 602

+ 1.84566 0 2 tcp 1000 - 2 0.1 3.2 102 611

- 1.84566 0 2 tcp 1000 - 2 0.1 3.2 102 611

+ : enqueue

- : dequeue

d : drop

r : get

The fields in the follow document are: kind of the occasion, recreation time when the occasion happened, source and objective hubs, bundle type (convention, activity or traffic source), parcel size, banners, stream id, source and objective locations, grouping number and bundle id. Notwithstanding following all occasions of the reproduction, it is additionally conceivable to make a follow object between a specific source and an objective with the order:

\$ns make follow <type> <file> <src> <dest>

where the sort can be, for example,

- Enque - a parcel appearance (for example at a line)
- Deque - a bundle flight (for example at a line)
- Drop - bundle drop

- Recv - bundle get at the objective

Following all occasions from a reenactment to a particular document and afterward figuring the ideal amounts from this record for example by utilizing perl or awk and Matlab is a simple way and appropriate when the geography is moderately straightforward and the quantity of sources is restricted. In any case, with complex geographies and numerous sources this method of gathering information can turn out to be excessively lethargic. The follow records will likewise burn-through a lot of circle space.

13.2.13 Monitors

With a line screen it is conceivable to follow the insights of appearances, flights and drops in either bytes or bundles. Alternatively the line screen can likewise keep a fundamental of the line size over the long haul.

For example, if there is a connection between hubs n0 and n1, the line screen can be set up as follows:

```
set qmon0 [$ns screen line $n0 $n1]
```

The bundle appearances and byte drops can be followed the orders:

```
set parr [$qmon0 set parrivals_]
```

```
set bdrop [$qmon0 set bdrops_]
```

Other than allotting an incentive to a variable the set order can likewise be utilized to get the estimation of a variable. For instance here the set order is utilized to get the estimation of the variable "parrivals" characterized in the line screen class.

A stream screen is like the line screen however it monitors the insights for a stream as opposed to for amassed traffic. A classifier initially figures out which stream the bundle has a place with and afterward passes the parcel to the stream screen.

The flow monitor can be made and joined to a specific connection with the orders:

```
set fm on [$ns make flowm on Fid]
```

```
$ns connect fm on [$ns interface $n1 $n3] $fm on
```

```
$fmon dump
```

On the off chance that you need to follow the insights for a specific stream, a classifier should be characterized so it chooses the stream dependent on its stream id, which could be for example 1:

```
set fclassifier [$fmon classifier]
```

```
set stream [$fclassifier query auto 0 1]
```

14.2.14 Controlling the recreation

After the reproduction geography is made, specialists are designed and so forth, the beginning and stop of the recreation and different occasions must be booked. The reenactment can be begun and halted with the orders

```
$ns at $simtime "finish"
```

```
$ns run
```

The principal order plans the method finish toward the finish of the reenactment, and the

subsequent order really begins the recreation. The completion strategy must be characterized to flush the follow support, close the follow documents and end the program with the leave schedule. It can alternatively begin NAM (a graphical organization illustrator), post cycle data and plot this data.

The completion method needs to contain at any rate the accompanying components:

```
proc finish {} {  
  
    worldwide ns trace_all  
  
    $ns flush-follow  
  
    close $trace_all  
  
    leave 0  
  
}
```

Different occasions, for example, the beginning or halting occasions of the customers can be planned for the accompanying way:

```
$ns at 0.0 "cbr0 start"  
  
$ns at 50.0 "ftp1start"  
  
$ns at $simtime "cbr0 stop"  
  
$ns at $simtime "ftp1 stop"
```

On the off chance that you have characterized your own systems, you can likewise plan the method to begin for instance at regular intervals in the accompanying manner:

```

proc model {} {

worldwide ns

set span 5

...

...

$ns at [expr $now + $interval] "model"

}

```

4.3 : Simple Stimulation Example

This segment presents a straightforward NS reenactment content and clarifies what each line does. Model 3 is an OTcl content which makes the basic organization arrangement and runs the reenactment situation in Figure 4. To run this recreation, duplicate the code in Example 3 of every a record named "ns-simple.tcl" and type "ns-simple.tcl" at shell brief

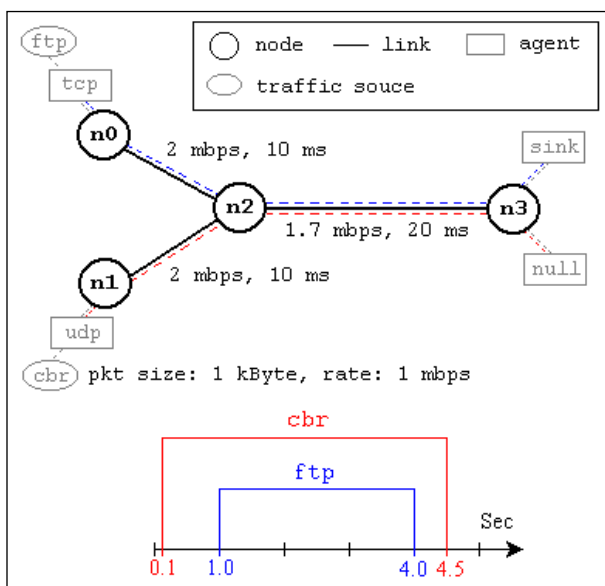


Figure 47. A Simple Network Topology and Simulation Scenario

The organization comprises of 4 hubs (n0, n1, n2, n3) as demonstrated in above figure. The duplex connections somewhere in the range of n0 and n2, and n1 and n2 have 2 Mbps of transmission capacity and 10 ms of postponement. The duplex connection somewhere in the range of n2 and n3 has 1.7 Mbps of transfer speed and 20 ms of deferral. Every hub utilizes a DropTail line, of which the most extreme size is 10. A "tcp" specialist is appended to n0, and an association is set up to a tcp "sink" specialist connected to n3. As default, the most extreme size of a bundle that a "tcp" specialist can create is 1KByte. A tcp "sink" specialist creates and sends ACK bundles to the sender (tcp specialist) and liberates the got parcels. A "udp" specialist that is joined to n1 is associated with a "invalid" specialist appended to n3. A "invalid" specialist simply liberates the bundles got. A "ftp" and a "cbr" traffic generator are appended to "tcp" and "udp" specialists separately, and the "cbr" is arranged to create 1 KByte bundles at the pace of 1 Mbps. The "cbr" is set to begin at 0.1 sec and stop at 4.5 sec, and "ftp" is set to begin at 1.0 sec and stop at 4.0 sec.

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
set nf [open out.nam w]
$ns nam trace-all $nf

#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
#Close the NAM trace file
```

```

        close $nf
        #Execute NAM on the trace file
        exec nam out.nam &
        exit 0
    }

#Create four nodes
    set n0 [$ns node]
    set n1 [$ns node]
    set n2 [$ns node]
    set n3 [$ns node]

#Create links between the nodes
    $ns duplex-link $n0 $n2 2Mb 10ms DropTail
    $ns duplex-link $n1 $n2 2Mb 10ms DropTail
    $ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10
    $ns queue-limit $n2 $n3 10

#Give node position (for NAM)
    $ns duplex-link-op $n0 $n2 orient right-down
    $ns duplex-link-op $n1 $n2 orient right-up
    $ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)
    $ns duplex-link-op $n2 $n3 queuePos 0.5

#Setup a TCP connection
    set tcp [new Agent/TCP]
    $tcp set class_ 2

```

```
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
```

#Setup a FTP over TCP connection

```
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
```

#Setup a UDP connection

```
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2
```

#Setup a CBR over UDP connection

```
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
```

#Schedule events for the CBR and FTP agents

```
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
```

```

$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Detach tcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3
$sink"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

#Run the simulation
$ns run

```

Example 3. A Simple NS Simulation Script

Coming up next is the clarification of the content above. By and large, a NS content beginnings with making a Simulator object example.

- • set ns [new Simulator]: creates a NS test system object occasion, and appoints it to variable ns (italics is utilized for factors and qualities in this segment). What this line does is the accompanying:
 - o Create a scheduler (default is schedule scheduler)
 - The "Test system" object has part works that do the accompanying:
 - o Create compound items like hubs and connections (depicted later)
 - o Connect network part protests made (ex. join specialist)
 - o Set network segment boundaries (generally for compound articles)
 - o Create associations between specialists (ex. make association between a "tcp" and "sink")
 - o Specify NAM show choices
 - o Etc.

- A large portion of part works are for reenactment arrangement and booking, anyway some of them are for the NAM show. The "Test system" object part work usage are situated in the "ns-2/tcl/lib/ns-lib.tcl" document's color fid color: is to set color of the packets for a flow specified by the flow id (fid). This is for the NAM display, and has no effect on the actual simulation.
- \$ns namtrace-all document descriptor: It advises the test system to record reenactment follows in NAM input design. It additionally gives the document name that the follow will be kept in touch with later by the order \$ns flush-follow. Likewise, the part work follow everything is for recording the recreation follow in an overall configuration.
- proc finish {}: is called after this reproduction is more than by the order \$ns at 5.0 "finish". In this capacity, post-reenactment measures are indicated.
- set n0 [\$ns node]: The part work hub makes a hub.
- \$ns duplex-interface node1 node2 transmission capacity defer line type: makes two simplex connections of indicated transfer speed and delay, and associates the two determined hubs. In NS, the yield line of a hub is executed as a piece of a connection, in this manner clients ought to determine the line type when making joins. In the above reenactment content, DropTail line is utilized.
- \$ns line limit node1 node2 number: This line sets the line furthest reaches of the two simplex connections that associate node1 and node2 to the number indicated.
- \$ns duplex-connect operation node1 node2: The several lines are utilized for the NAM show. This line of code is utilized to screen the line for the connection between the hubs node1 and node2. Client can see the parcels in the line, and inevitably the client can even perceive how the bundles are being dropped.

Since the fundamental organization arrangement is done, the following activity is to arrangement traffic specialists like TCP and UDP, traffic sources like FTP and CBR, and append them to hubs and specialists individually.

- `set tcp [new Agent/TCP]`: This line makes a TCP specialist. To make specialists or traffic sources, a client should realize the class names these items (Agent/TCP, Agent/TCPSink, Application/FTP, etc).

- `$ns join specialist hub specialist`: The `append specialist` part work connects a specialist object made to a hub object. As a matter of fact, this capacity calls the `append` part capacity of indicated hub, which connects the offered specialist to itself. Hence, a client can do something very similar by, for instance, `$n0 connect $tcp`.

- `$ns interface agent1 agent2`: After two specialists that will speak with one another are made, the following thing is to set up a consistent organization association between them.

Accepting that all the organization design is done, the following activity is compose a recreation situation (for example reenactment planning). The Simulator object has many booking part works. Notwithstanding, the one that is generally utilized is the accompanying:

- `$ns at time "string"`: This part capacity of a Simulator object makes the scheduler (`scheduler_` is the variable that focuses the scheduler object made by `[new Scheduler]` order toward the start of the content) to plan the execution of the predefined string at given recreation time. For instance, `$ns at 0.1 "$cbr start"` will settle on the scheduler decision a beginning part capacity of the CBR traffic source object, what begins the CBR to communicate information.

After all organization setups, booking and post-reenactment system determinations are done, the solitary thing left is to run the recreation. This is finished by `$ns run`.

4.4 Extending NS: Creating another Agent

One can broaden NS by adding new conventions. This segment will examine about how another specialist can be made in NS utilizing a model. The code in this segment

executes a type of straightforward 'ping' convention. One hub will send n client characterized number bundles, each in turn at standard stretches, to another hub which will restore the parcel right away. For every bundle the sender will at that point ascertain the full circle time.

4.5 The header record

First we need to make a header document "myPing.h" in which we will characterize the information structure for the myPing parcel header.

```
struct hdr_myPing {
    char ret;
    double send_time;

    // Header access methods
    static int offset_; // required by PacketHeaderManager
    inline static int& offset() { return offset_; }
    inline static hdr_myPing* access(const Packet* p) {
        return (hdr_myPing*) p->access(offset_);
    }
};
```

The singe 'ret' will be set to '0' if the parcel is on its way from the sender to the hub which is being pinged, while it will be set to '1' on its way back. The twofold 'send time' is a period stamp that is determined to the bundle when it is sent. This time is subsequently utilized by the sender to ascertain the full circle time. The remainder of the piece of program codes of the header are utilized to get to the bundle header from any parcel reference.

The accompanying piece of code pronounces the class 'My Ping Agent' as a subclass of the class 'Specialist'.

```

class MyPingAgent : public Agent {
public:
    MyPingAgent();
    int command(int argc, const char*const* argv);
    void recv(Packet*, Handler*);
    void send_packets();
protected:
    int num_packets_;
    double wait_interval_;
    MyPingAgentTimer timer_;
    int count;
};

```

The num parcels variable holds the complete number of bundles that will be sent by the sender and 'stand by stretch' is the time span between two progressive parcels. The variable 'tally' keeps the check of parcels sent up until now. We additionally need to characterize a clock controller (My Ping Agent Timer) for the sender with the goal that it can reschedule itself each time subsequent to sending a bundle. My Ping Agent will utilize a variable 'clock' of type My Ping Agent Timer to monitor time. The source code for the header record "myPing.h" is given beneath:

```

/* myPing.h */

#ifndef ns_myPing_h
#define ns_myPing_h

#include "agent.h"
#include "tclcl.h"
#include "packet.h"
#include "address.h"
#include "ip.h"

```

```

class MyPingAgent;

struct hdr_myPing {
    char ret;
    double send_time;

    // Header access methods
    static int offset_; // required by PacketHeaderManager
    inline static int& offset() { return offset_; }
    inline static hdr_myPing* access(const Packet* p) {
        return (hdr_myPing*) p->access(offset_);
    }
};

class MyPingAgentTimer : public TimerHandler {
public:
    MyPingAgentTimer(MyPingAgent *t) : TimerHandler()
        { t_ = t; }

protected:
    virtual void expire(Event *e);
    MyPingAgent *t_;
};

class MyPingAgent : public Agent {
public:
    MyPingAgent();
    int command(int argc, const char*const* argv);
    void recv(Packet*, Handler*);
    void send_packets();
protected:
    int num_packets_;
    double wait_interval_;
};

```

```

        MyPingAgentTimer timer_;
        int count;
};

#endif

```

Table 1. MyPing.h

- o The C++ code for new Agent

First the linkage between the C++ code and Tcl code must be characterized.

```

static class kangAbHeaderClass : public PacketHeaderClass {
public:
    kangAbHeaderClass() : PacketHeaderClass
        ("PacketHeader/MyPing", sizeof(hdr_myPing)) {
        bind_offset(&hdr_myPing::offset_);
    }
} class_kangAbhdrSnd;

static class MyPingAgentClass : public TclClass {
public:
    MyPingAgentClass() : TclClass("Agent/MyPing") {}
    TclObject* create(int, const char*const*) {
        return (new MyPingAgent());
    }
} class_myPing;

```

The following piece of code is the constructor for the class 'MyPingAgent'. It ties the factors which must be gotten to both in Tcl and C++.

```

MyPingAgent::MyPingAgent() : Agent(PT_MYPING), timer_(this) , count(0)
{
    bind("packetSize_", &size_);
    bind("num_packets_", &num_packets_);
    bind("wait_interval_" &wait_interval_);
}

```

```
}
```

The capacity 'order()' is considered when a Tcl order for the class 'My Ping Agent' is executed. For our situation that would be "\$sender send" (expecting 'sender' is an occurrence of the Agent/My Ping class), since we need to send ping parcels from the Agent to another ping specialist. We need to parse the order in the 'order()' work. On the off chance that the order matches, it will thus the 'send packets ()' work which sends each bundle in turn and reschedules itself again to send the following parcel after the stand by span. In the event that no match is discovered, it is passed to the to the 'order()' capacity of the base class (for this situation 'Agent::command()') with its contentions.

```
int MyPingAgent::command(int argc, const char*const* argv)
{
    if (argc == 2) {
        if (strcmp(argv[1], "send") == 0) {
            send_packets();
            // return TCL_OK, so the calling function knows that
            // the command has been processed
            return (TCL_OK);
        }
    }
    return (Agent::command(argc, argv));
}

void MyPingAgent::send_packets()
{
    if(count++ < num_packets_)
    {
        Packet *pkt = allocpkt();
        hdr_myPing *hdr = hdr_myPing::access(pkt);
        // Set the 'ret' field to 0, so the receiving node knows
        // that it has to generate an echo packet
        hdr->ret = 0;
    }
}
```

```

        // Store the current time in the 'send_time' field
        hdr->send_time = Scheduler::instance().clock();
        // Send the packet
        send(pkt, 0);
    }
    if(count < num_packets_)
        timer_.resched(wait_interval_);
}

```

The function 'recv()' defines the actions to be taken when a packet is received. If the 'ret' field is 0, a packet with the same value for the 'send time' field, but with the 'ret' field set to 1 has to be returned. If 'ret' is 1, the roundtrip time of the packet is calculated and printed. The complete source code for "myPing.cc" is given below:

```

/* myPing.cc */

#include "myPing.h"

int hdr_myPing::offset_;

static class kangAbHeaderClass : public PacketHeaderClass {
public:
    kangAbHeaderClass() :
        PacketHeaderClass("PacketHeader/MyPing",
                           sizeof(hdr_myPing)) {
        bind_offset(&hdr_myPing::offset_);
    }
} class_kangAbhdrSnd;

static class MyPingAgentClass : public TclClass {
public:
    MyPingAgentClass() : TclClass("Agent/MyPing") {}
}

```

```

TclObject* create(int, const char*const*) {
    return (new MyPingAgent());
}
} class_myPing;

/* constructor*/
MyPingAgent::MyPingAgent() : Agent(PT_MYPING), timer_(this), count(0)
{
    bind("packetSize_", &size_);
    bind("num_packets_", &num_packets_);
    bind("wait_interval_", &wait_interval_);
}

/* if timer expires just do a timeout */
void MyPingAgentTimer::expire(Event*)
{
    t_>send_packets();
}

int MyPingAgent::command(int argc, const char*const* argv)
{
    if (argc == 2) {
        if (strcmp(argv[1], "send") == 0) {
            send_packets();
            // return TCL_OK, so the calling function knows that
            // the command has been processed
            return (TCL_OK);
        }
    }
    return (Agent::command(argc, argv));
}

```

```

void MyPingAgent::send_packets()
{
    if(count++ < num_packets_)
    {
        Packet *pkt = allocpkt();
        hdr_myPing *hdr = hdr_myPing::access(pkt);
        // Set the 'ret' field to 0, so the receiving node knows
        // that it has to generate an echo packet
        hdr->ret = 0;
        // Store the current time in the 'send_time' field
        hdr->send_time = Scheduler::instance().clock();
        // Send the packet
        send(pkt, 0);
    }
    if(count < num_packets_)
        timer_.resched(wait_interval_);
}

void MyPingAgent::recv(Packet* pkt, Handler*)
{
    hdr_myPing *hdr=hdr_myPing::access(pkt);
    if (hdr->ret == 0) {
        // Send an 'echo'. First save the old packet's send_time
        double stime = hdr->send_time;
        // Discard the packet
        Packet::free(pkt);
        // Create a new packet
        Packet* pktret = allocpkt();
        // Access the Ping header for the new packet:
        hdr_myPing* hdrret = hdr_myPing::access(pktret);
        // Set the 'ret' field to 1
        hdrret->ret = 1;
    }
}

```

```

        // Set the send_time field to the correct value
        hdrret->send_time = stime;
        // Send the packet
        send(pktret, 0);
    }
    else {
        // Get the receiving time of the packet
        double rcv_time = Scheduler::instance().clock();
        printf("RTT = %lf\n",rcv_time-hdr->send_time);
        // Discard the packet
        Packet::free(pkt);
    }
}

```

Table 2. myPing.cc

Presently save these two records "myPing.h" and "myPing.cc" in the "... ./ns-allinone-2.31/ns-2.31/" envelope. For great practice make a different index for all projects for client characterized conventions. In our model we have kept these two records in "... ./ns-allinone-2.31/ns-2.31/bwtools/MyPing" registry.

4.6 Necessary changes in NS

Subsequent stage is to adjust a portion of the NS framework records so the recently composed convention or specialist can work with the test system. The records that are should have been adjusted are appeared inside red air pockets.

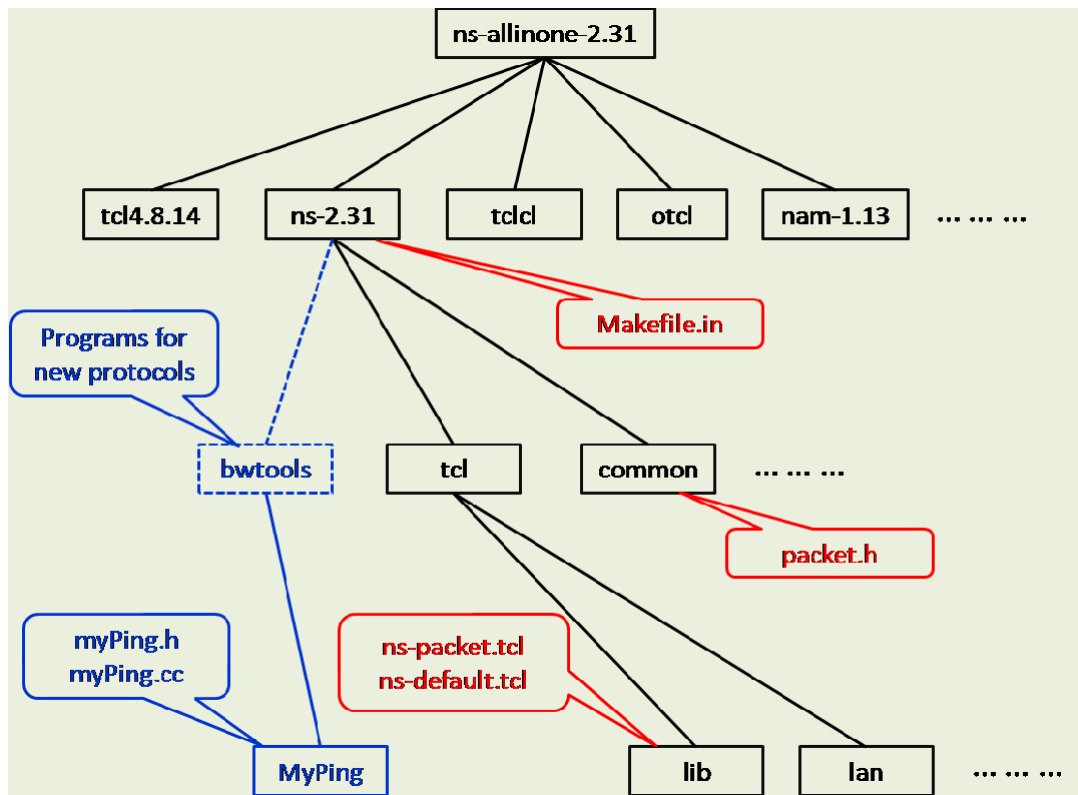


Fig 4.8 Figure of NS directory structure

5. RESULTS AND DISCUSSION

5.1 Edit packet.h file

We need another parcel type for our MyPing specialist, in this way the initial step is to adjust the "\$NS/normal/packet.h" record (expecting that the \$NS climate variable focuses the/ns-allinone-2.31/ns-2.31 area). There you can discover the definitions for the bundle convention IDs (for example PT_TCP, PT_TELNET, and so forth) Add another definition for PT_PING there. In my altered variant of packet.h, the last couple of lines of enum packet_t{} resembles the accompanying code (it may look somewhat changed in changed NS discharges).

```
enum packet_t {
PT_TCP,
PT_UDP,
.....
// insert new packet types here
PT_TFRC,
PT_TFRC_ACK,
PT_MYPING, // packet protocol ID for MyPing agent
PT_NTTYPE // This MUST be the LAST one
};
```

You also have to edit the **p_info()** in the same file to include "MyPing".

```
class p_info {
public:
p_info() {
name_[PT_TCP]= "tcp";
name_[PT_UDP]= "udp";
.....
}
```

```

name_[PT_TFRC]= "tcpFriend";
name_[PT_TFRC_ACK]= "tcpFriendCtl";

name_[PT_MYPING]="MyPing";

name_[PT_NTTYPE]= "undefined";
}
.....
}

```

5.2 Edit ns-packet.tcl document

Presently in "\$NS/tcl/lib/ns-packet.tcl" record look for the part `foreach prot { .. }`. Inside this part add a passage for MyPing. It will look like after:

```

foreach prot {
# Common:
    Common
    Flags
    IP      # IP
    ... ..
    Pushback  # Pushback, router-to-router

    MyPing

# Multicast:
    aSRM      # mcast/srm.cc

```

```

... ..
# Other:
    Encap      # common/encap.cc
    IPinIP    # IP encapsulation
    HDLC      # High Level Data Link Control
}{
    add-packet-header $prot
}

```

5.3 Edit ns-default.tcl

The “\$NS/tcl/lib/ns-default.tcl” file is the file where all default values for the Tcl objects are defined. Insert the following line to set the default packet size, packet number and wait interval for Agent/MyPing.

```

Agent/MyPing set packetSize_ 200
Agent/MyPing set num_packets_ 3
Agent/MyPing set wait_interval_ .001

```

5.4 Edit Makefile.in

The last change is a change that has to be applied to the “\$NS/Makefile.in”. You have to add the file “myPing.o” to the list of object files for ns. In my version the last lines of the edited list look like this:

```

OBJ_CC = \
    tools/random.o tools/rng.o tools/ranvar.o common/misc.o common/timer-
handler.o \
    common/scheduler.o common/object.o common/packet.o \
    ... ..

```

```
bwtools/MyPing/myPing.o \  
tools/trafgen.o trace/traffictrace.o tools/pareto.o \  
... ..
```

5.5 Recompile NS

After the changes are done in the NS files we need to recompile the NS simulator. To do so use the following command:

```
droy@aggarp3:~$ cd $NS  
droy@aggarp3:ns-2.31$ pwd  
/usr/share/ns-allinone-2.31/ns-2.31  
droy@aggarp3:ns-2.31$ sudo ./configure  
droy@aggarp3:ns-2.31$ sudo make
```

5.6 Simple example using MyPing agent

After recompiling the NS simulator we can use our MyPing agent in our simulations. Following code is simple tcl program that uses the MyPing agent.

```
set ns [new Simulator]  
set nf [open out.tr w]  
$ns trace-all $nf  
  
#Define a 'finish' procedure  
proc finish {} {  
    global ns nf  
    $ns flush-trace  
    #Close the trace file  
    close $nf  
    #Execute nam on the trace file  
    exit 0  
}
```

```

}
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 10Mb 50ms DropTail

#####Sender Side Stuff#####
set sender [new Agent/MyPing]
$ns attach-agent $n0 $sender
$sender set packetSize_ 1500
$sender set num_packets_ 5
$sender set wait_interval_ .05

#####Receiver Side Stuff
set recv [new Agent/MyPing]
$ns attach-agent $n1 $recv

#####Connect the sender with receiver
$ns connect $sender $recv

#Schedule events for the CBR agent
$ns at 0.0 "$sender send"
$ns at 10.0 "finish"
$ns run

```

Figure 1. myPingExample.tcl

5.7 AES – NAM Results

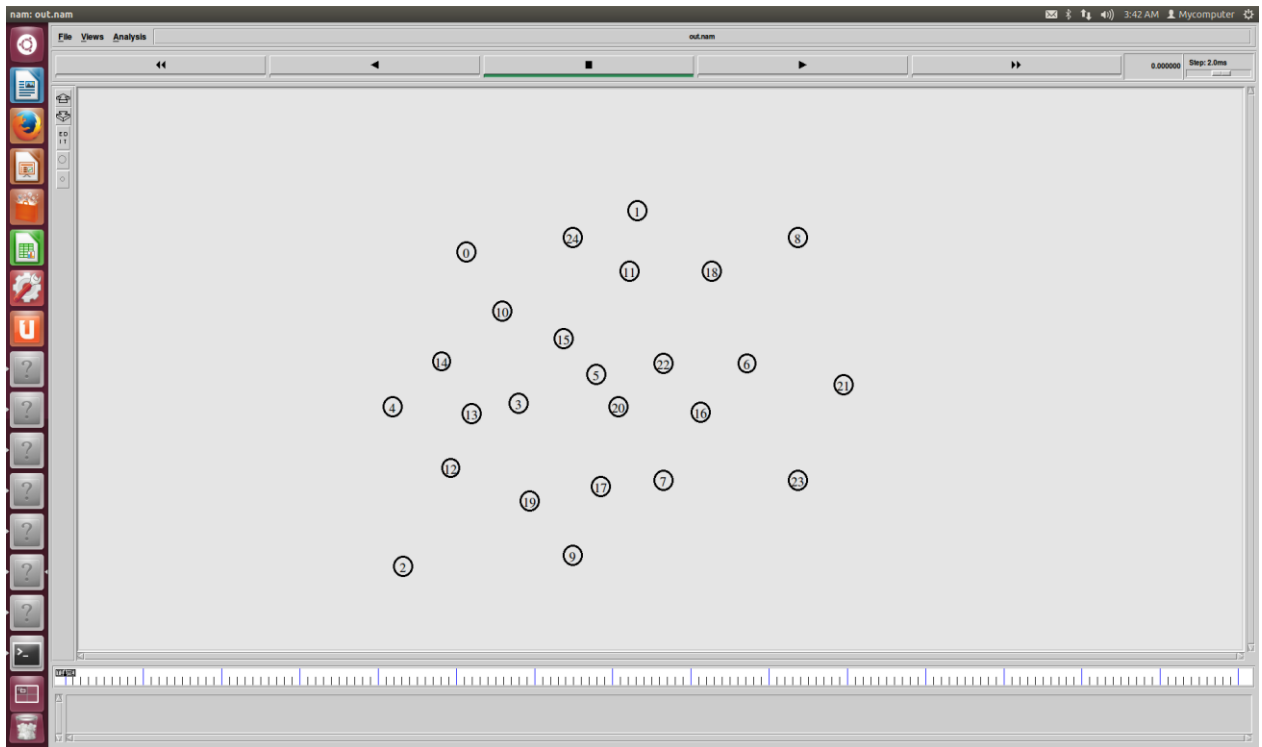
Algorithm: AES (Advanced Encryption Standard)

Input Data : person_70_120

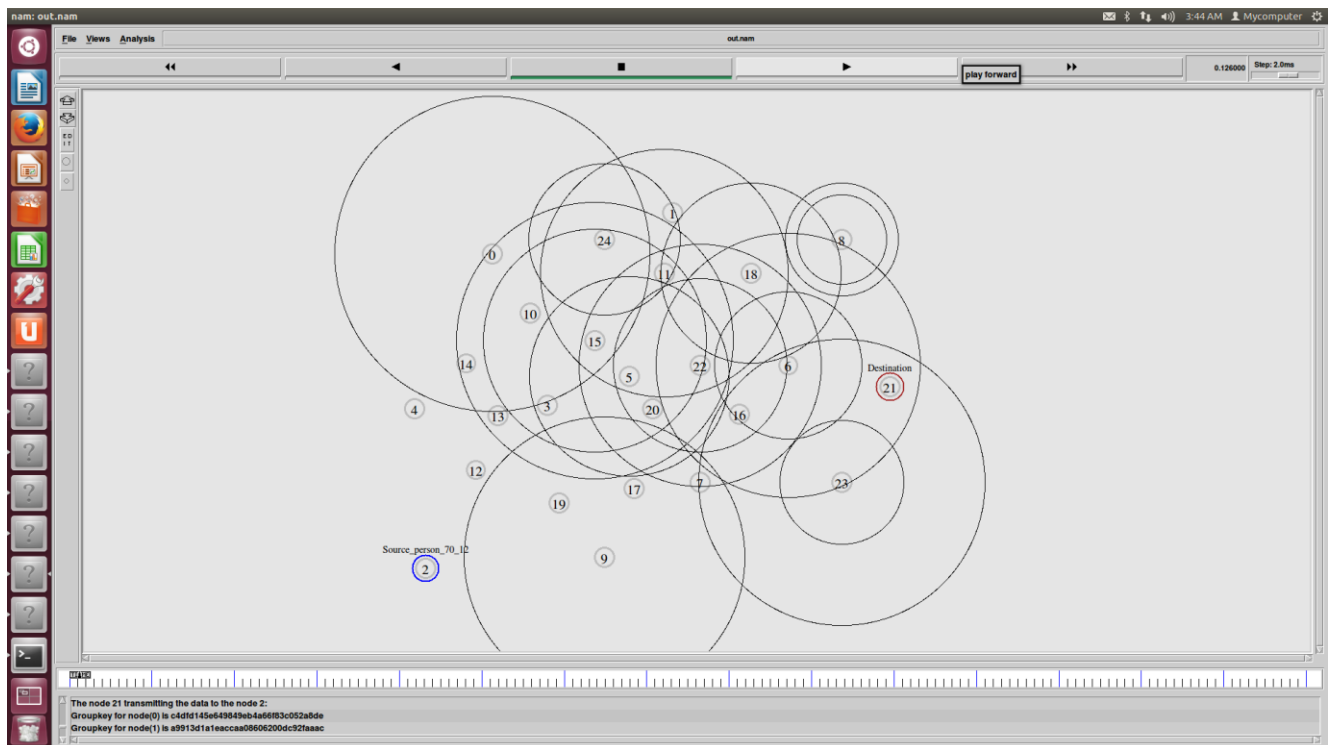
Transmitter node: 2

Receiver node: 21

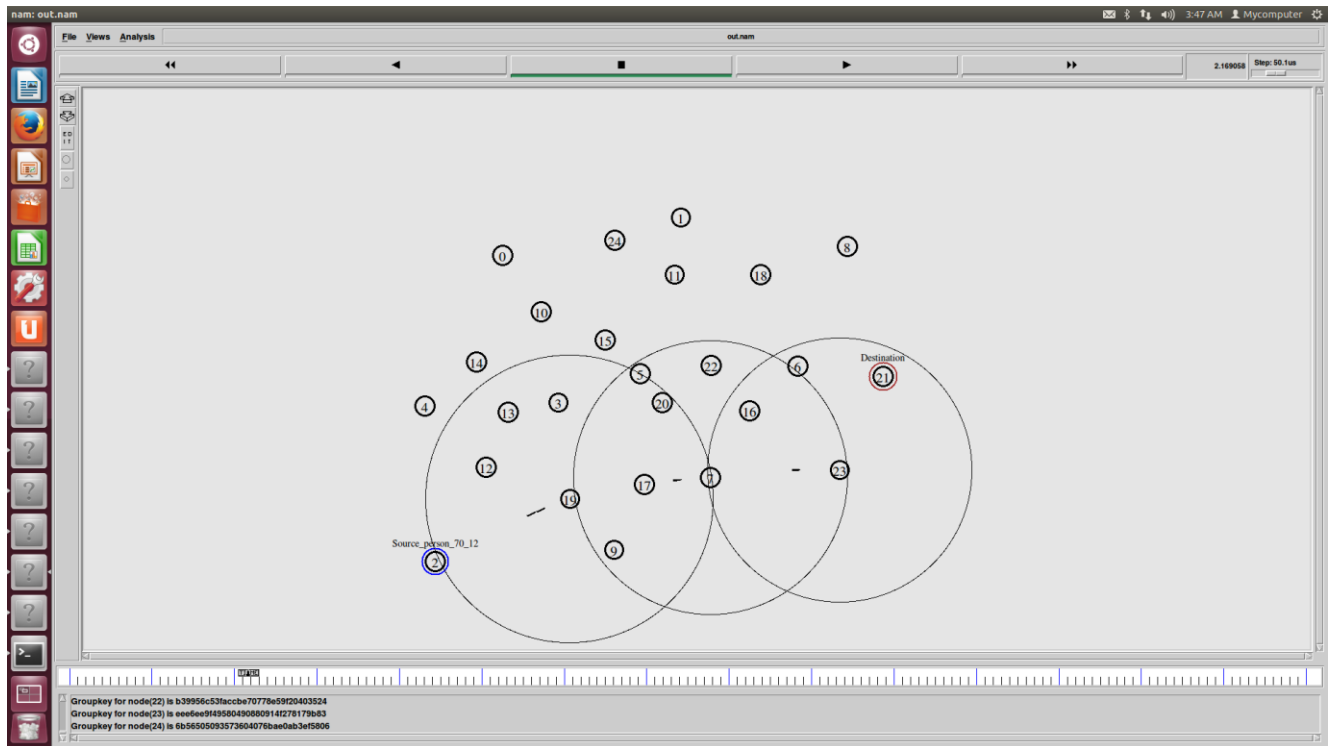
Total nodes: 24



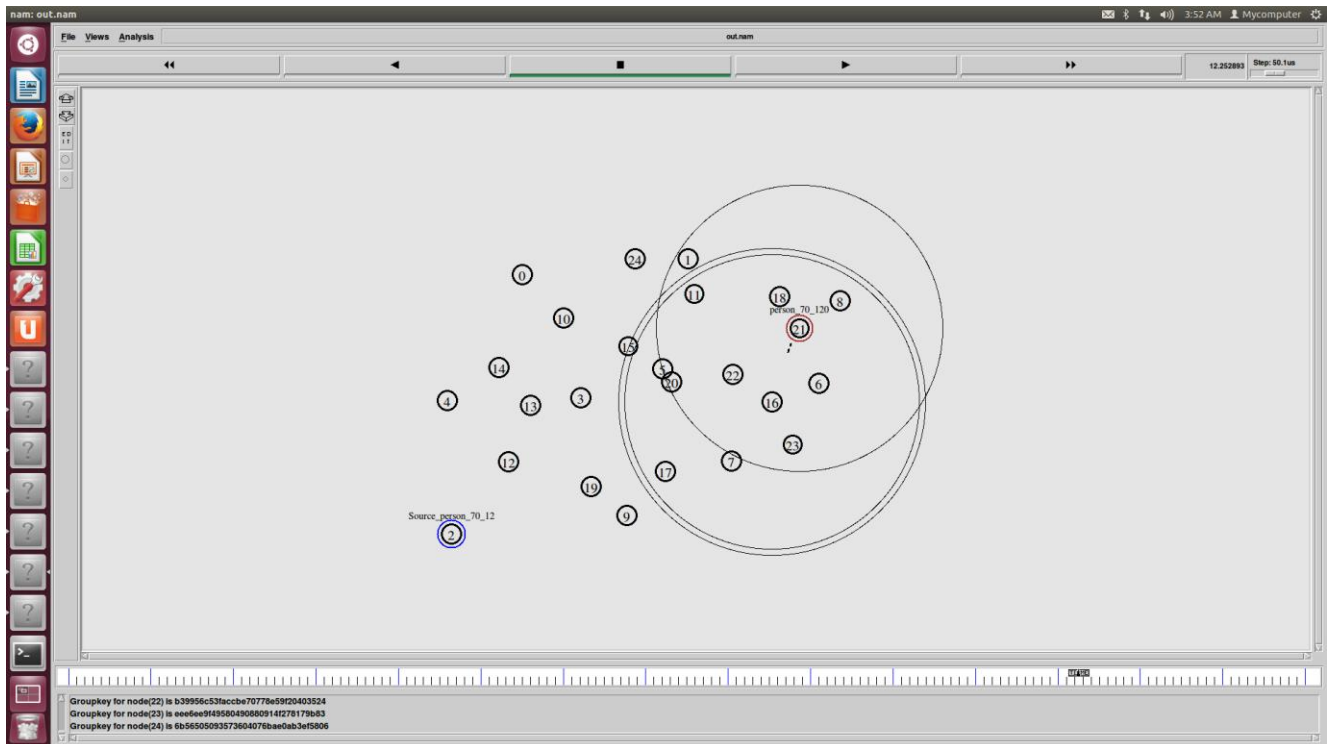
5.1 Network Representation NAM output file (AES)



5.2 Authenication between Nodes

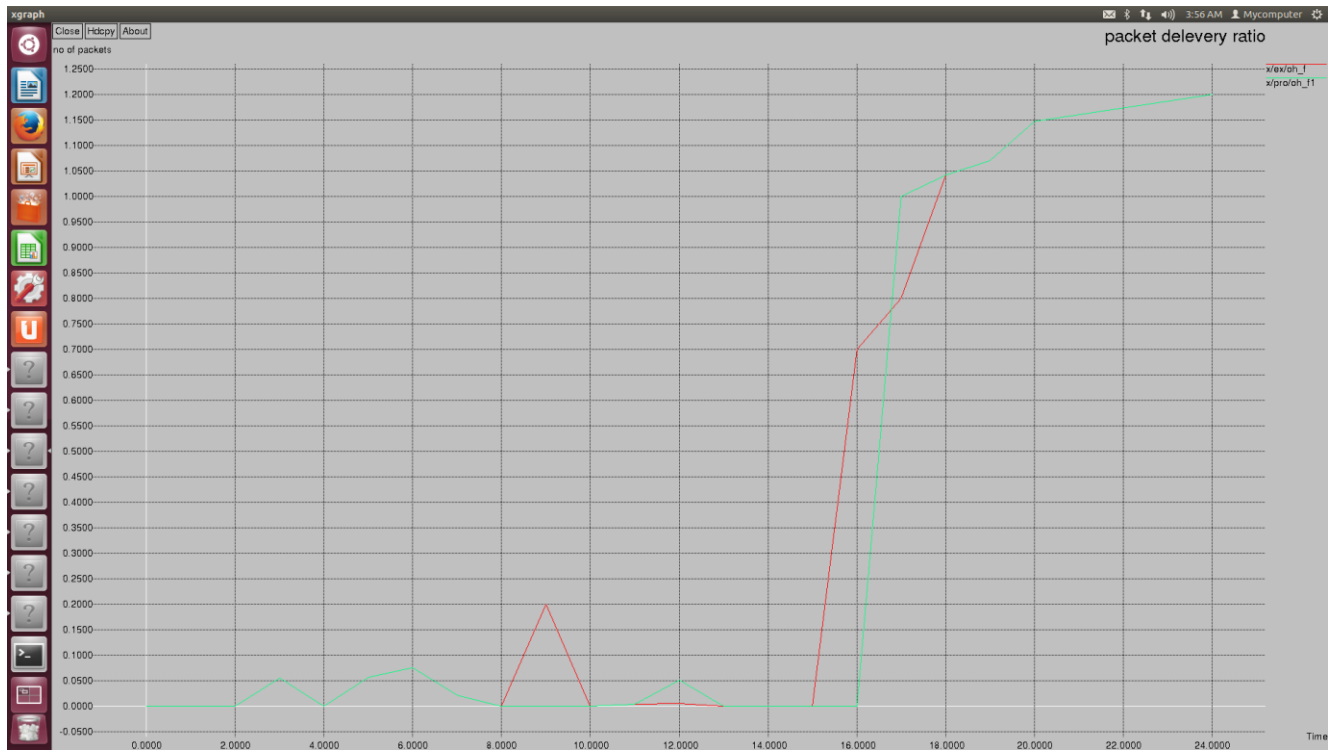


5.3 Routing in Network

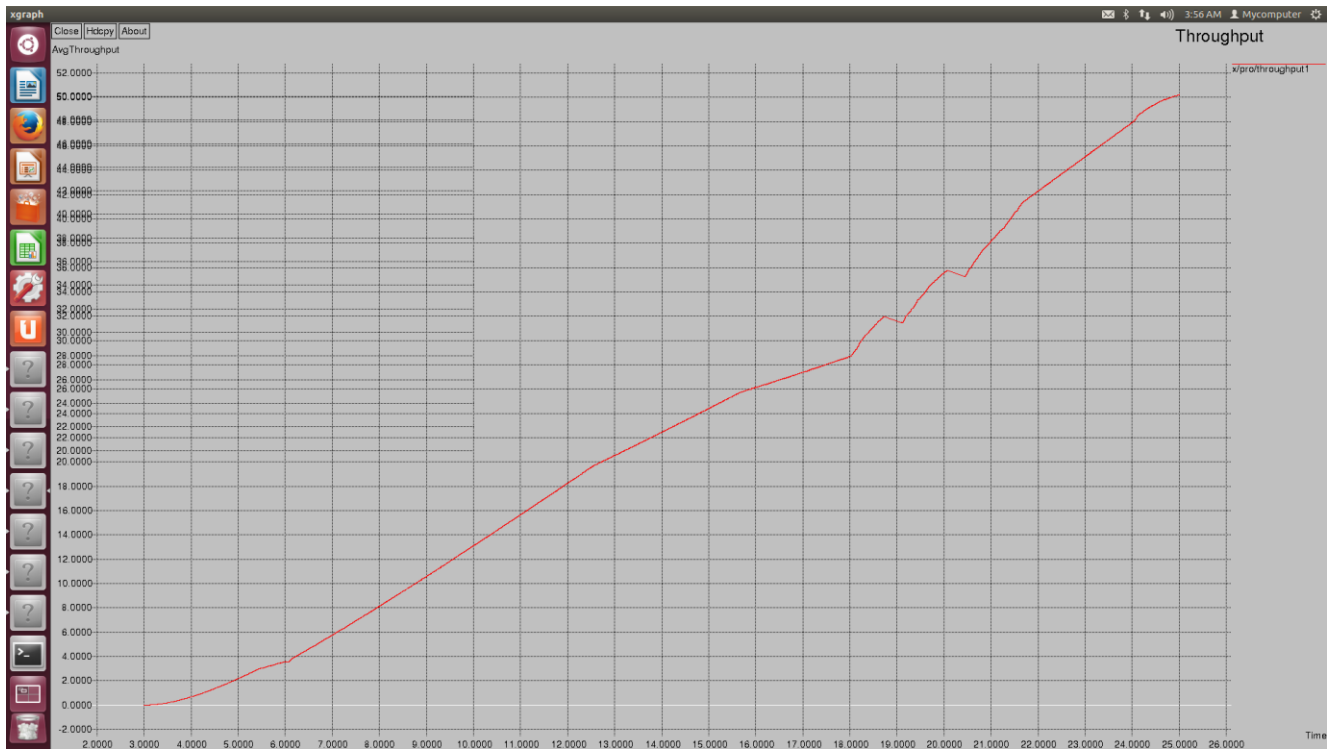


5.4 Receiving data at Receiver node from transmitter node

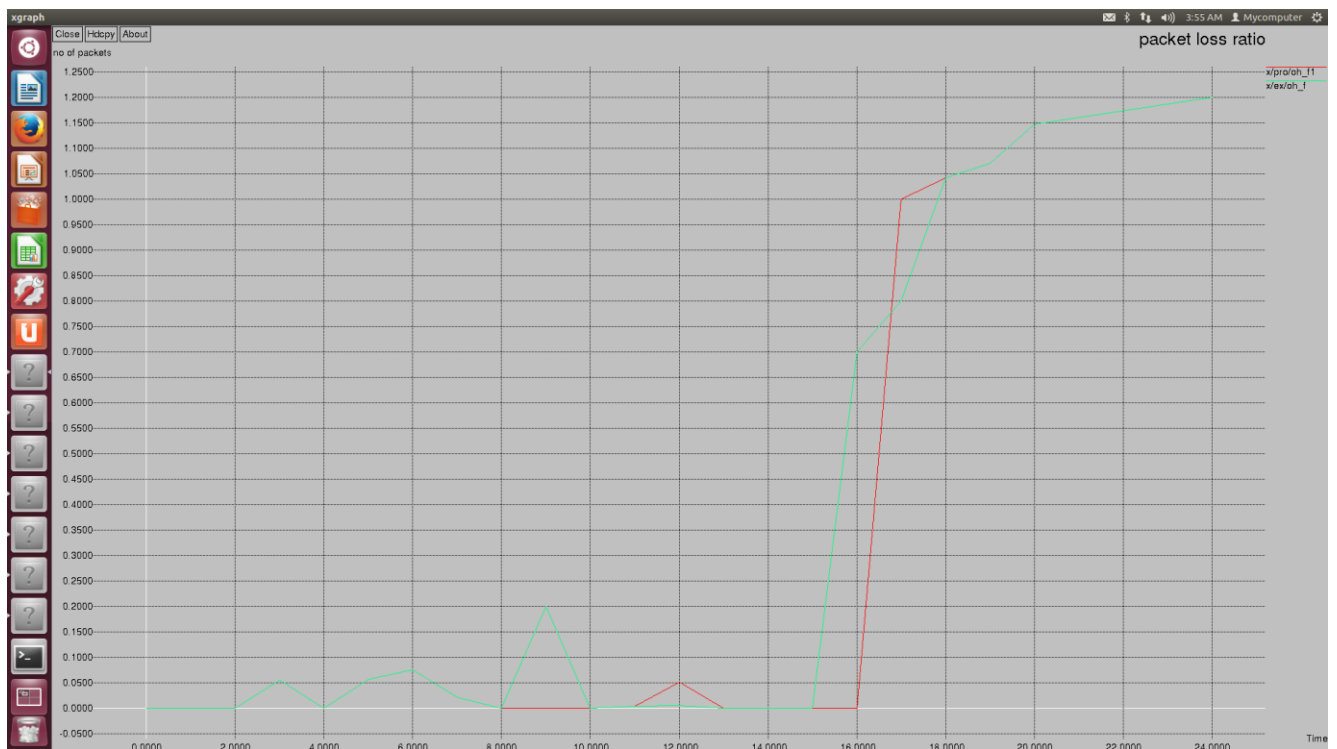
5.6 AES Parameter Analysis



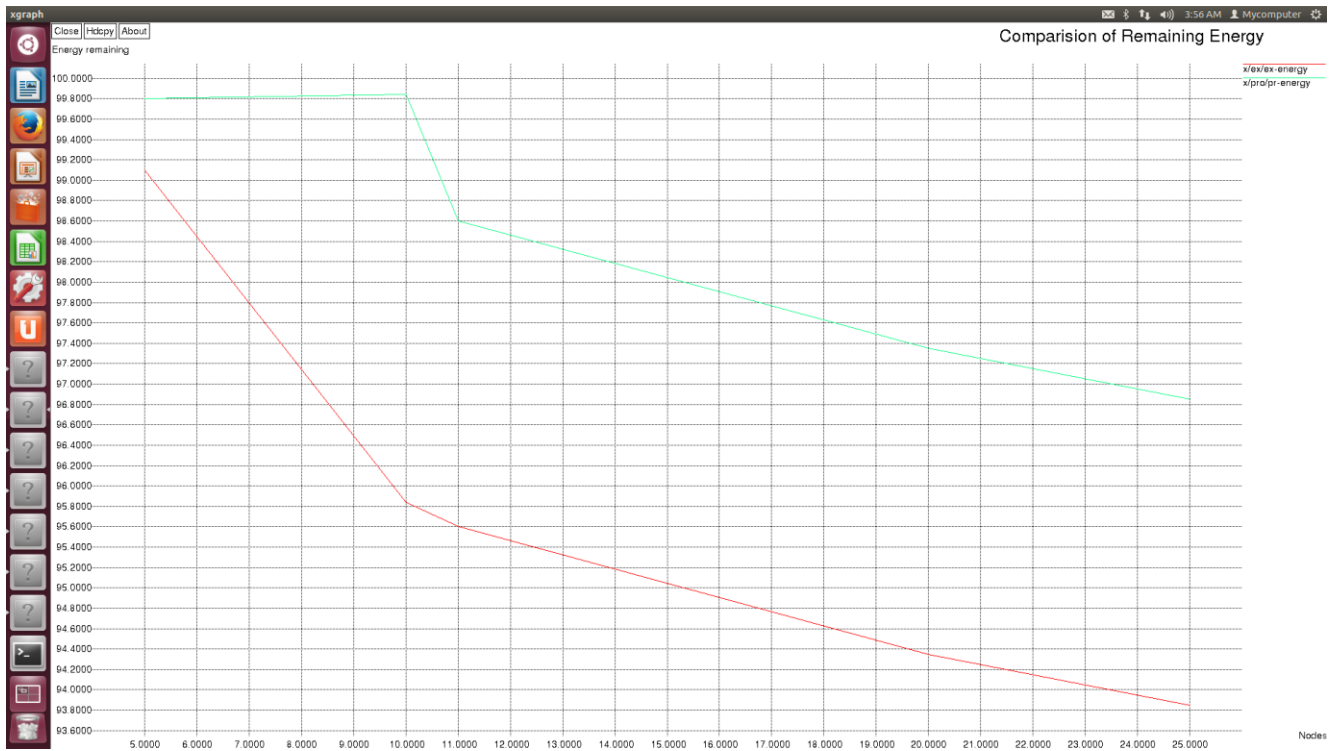
5.9 (a) Packet Delivery Ratio



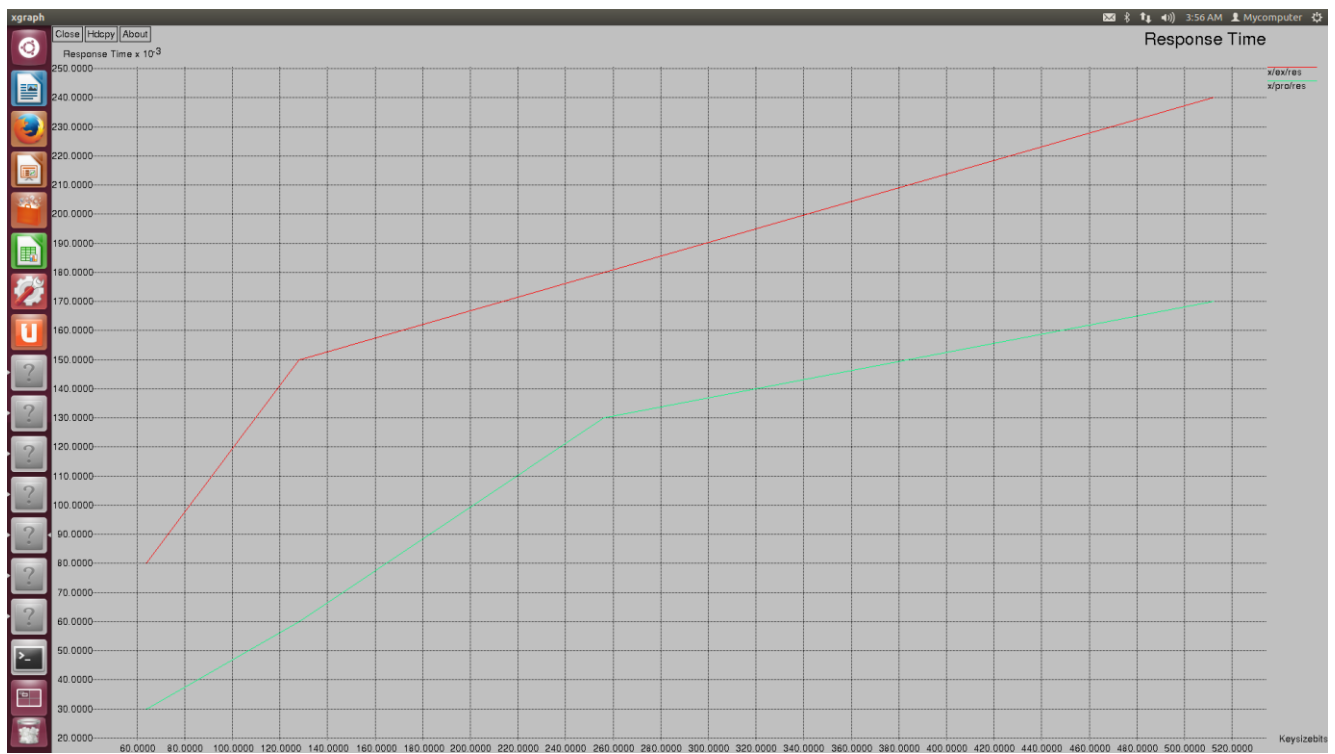
5.9(b) Throughput



5.9(c) Packet Loss Ratio



5.9(d) Comparison Of Energy



5.9(e) Response Time

5.8 RC4 - NAM Results

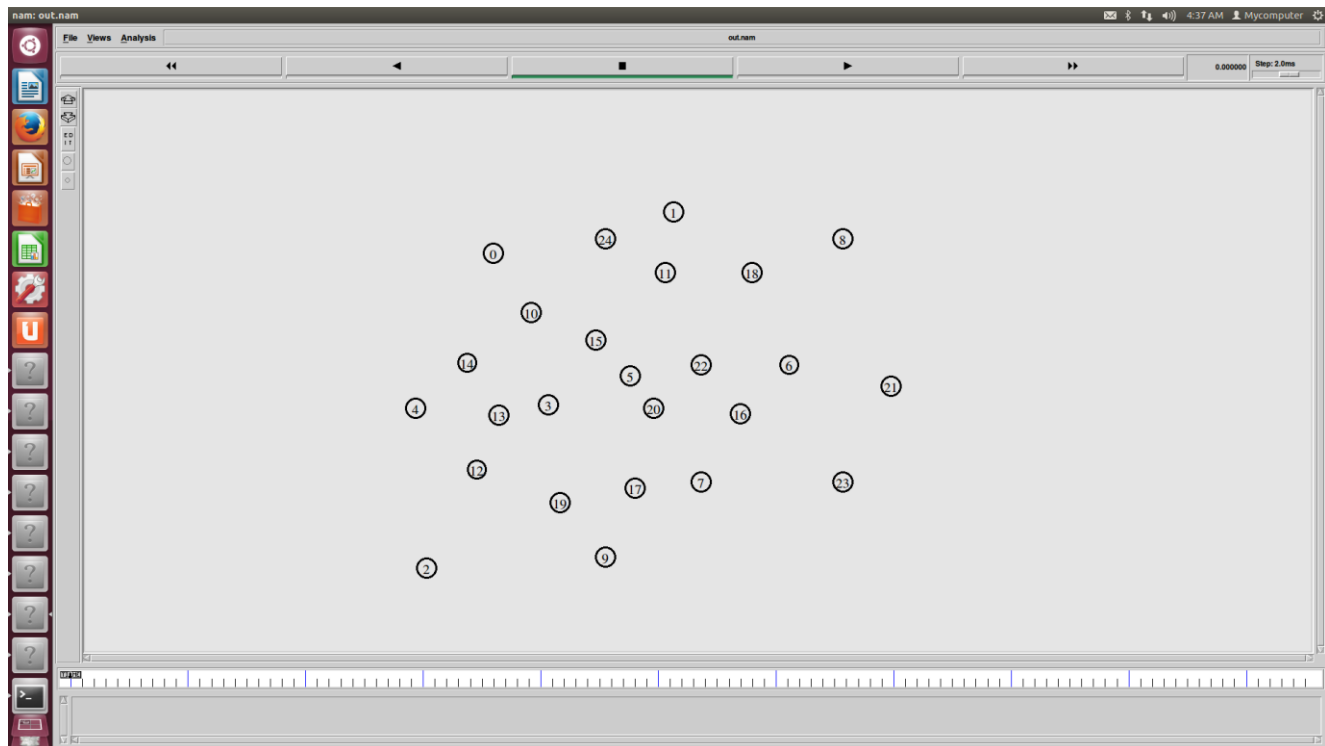
Algorithm: RC4 (Rivest Cipher 4)

Input Data: hello_2021

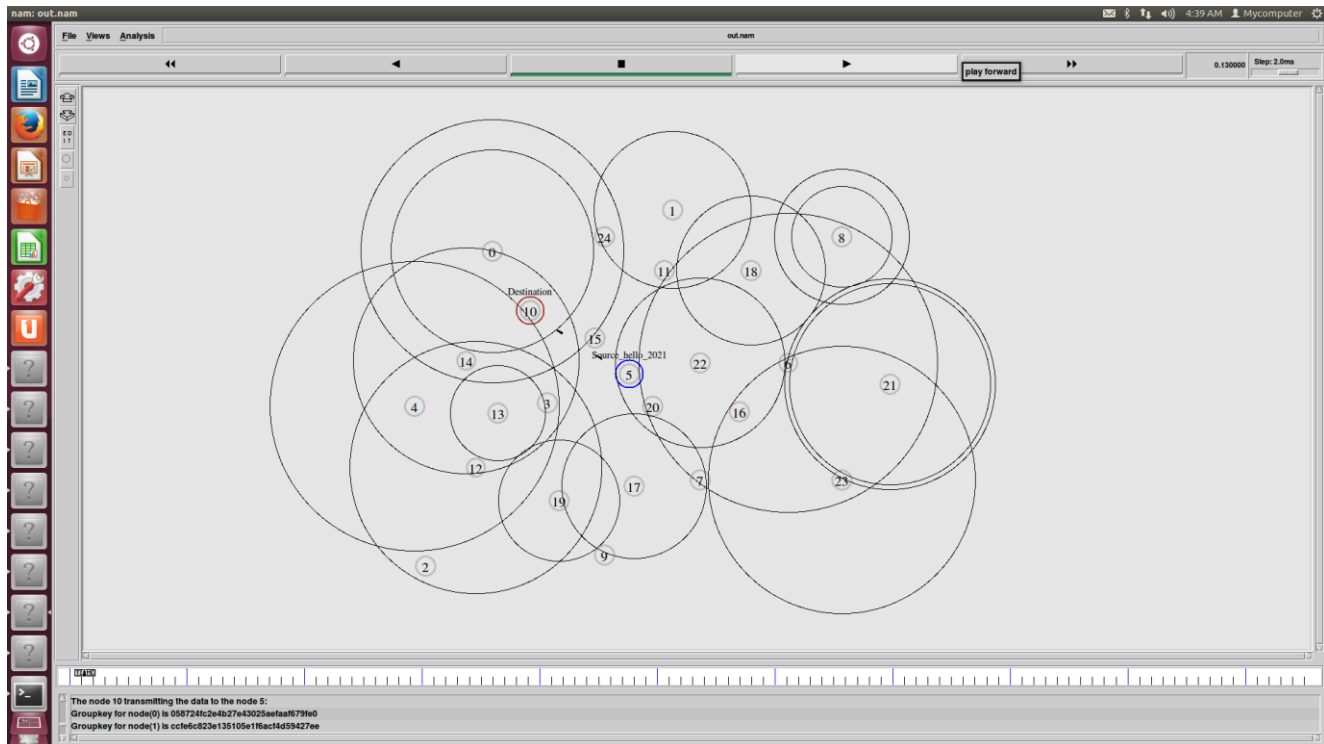
Transmitter node: 5

Receiver node: 10

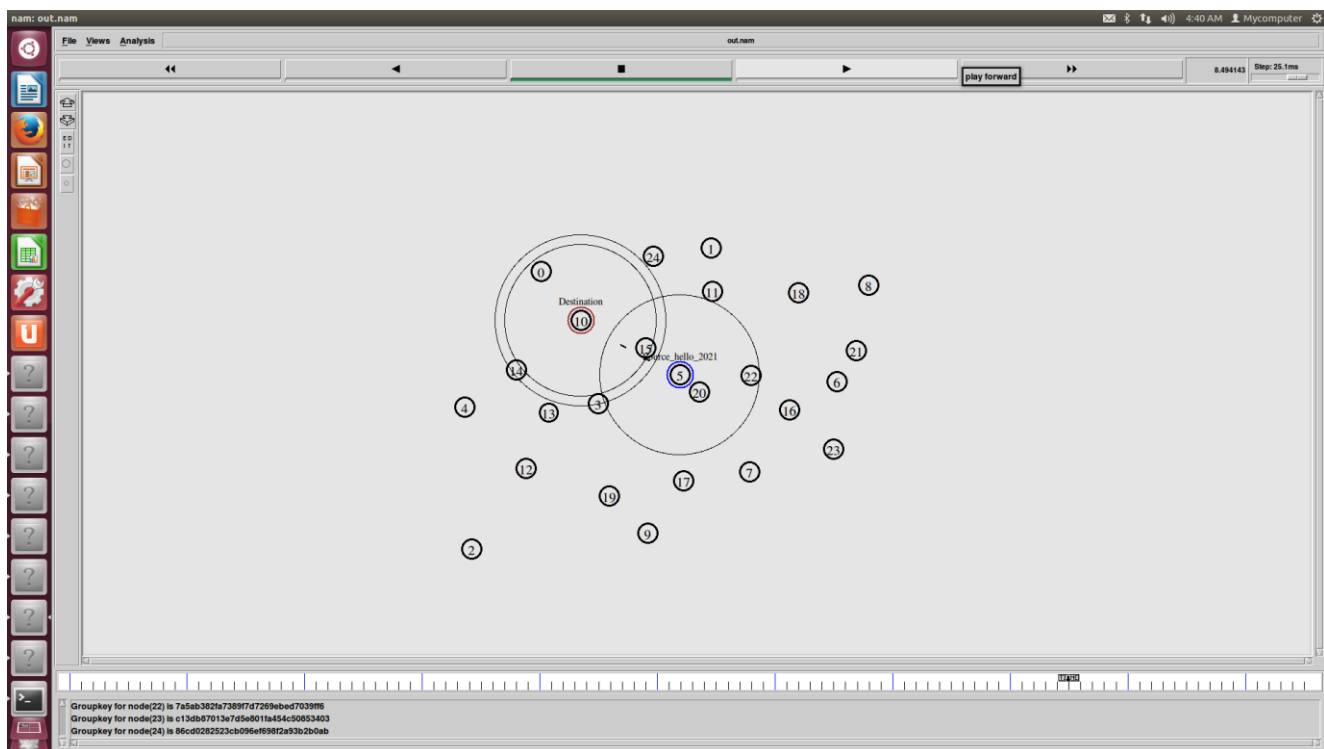
Total nodes: 24



5.9.1 Network Representation NAM output (RC4)

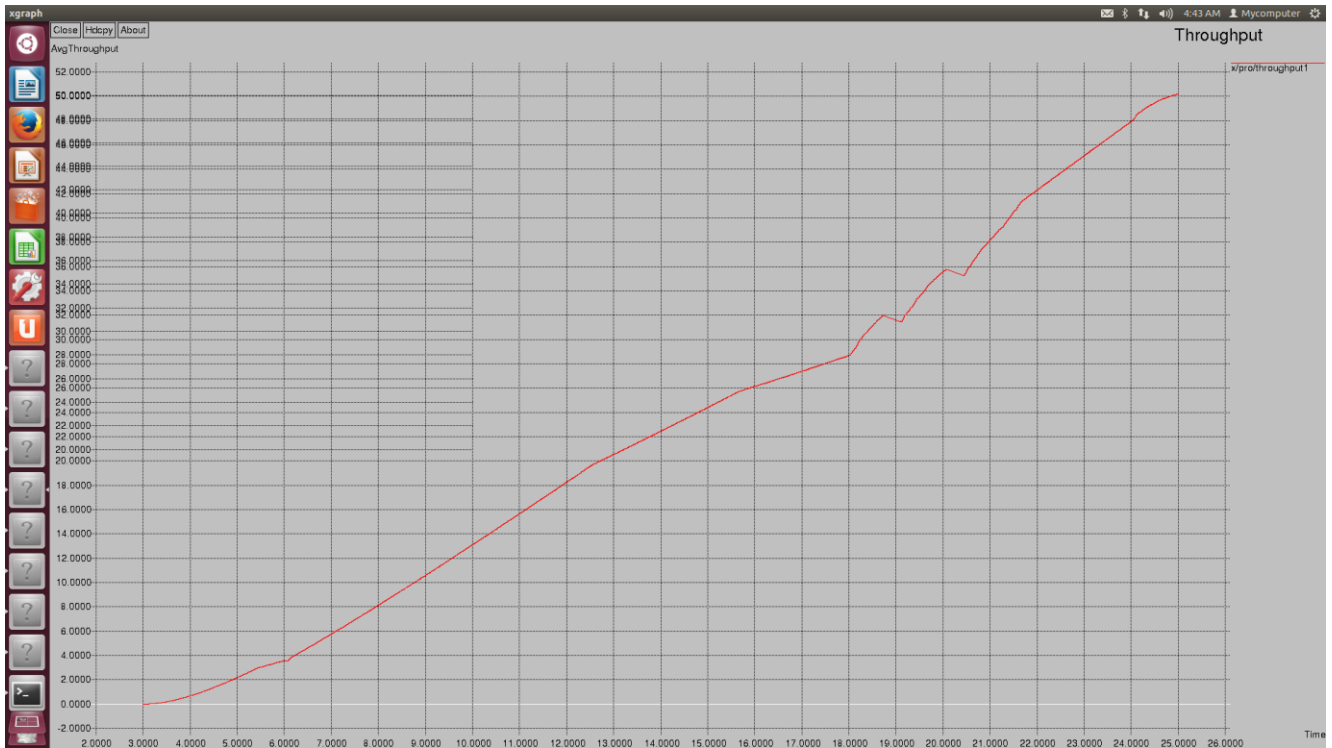


5.9.2 Authentication between Transmitter and Receiver nodes

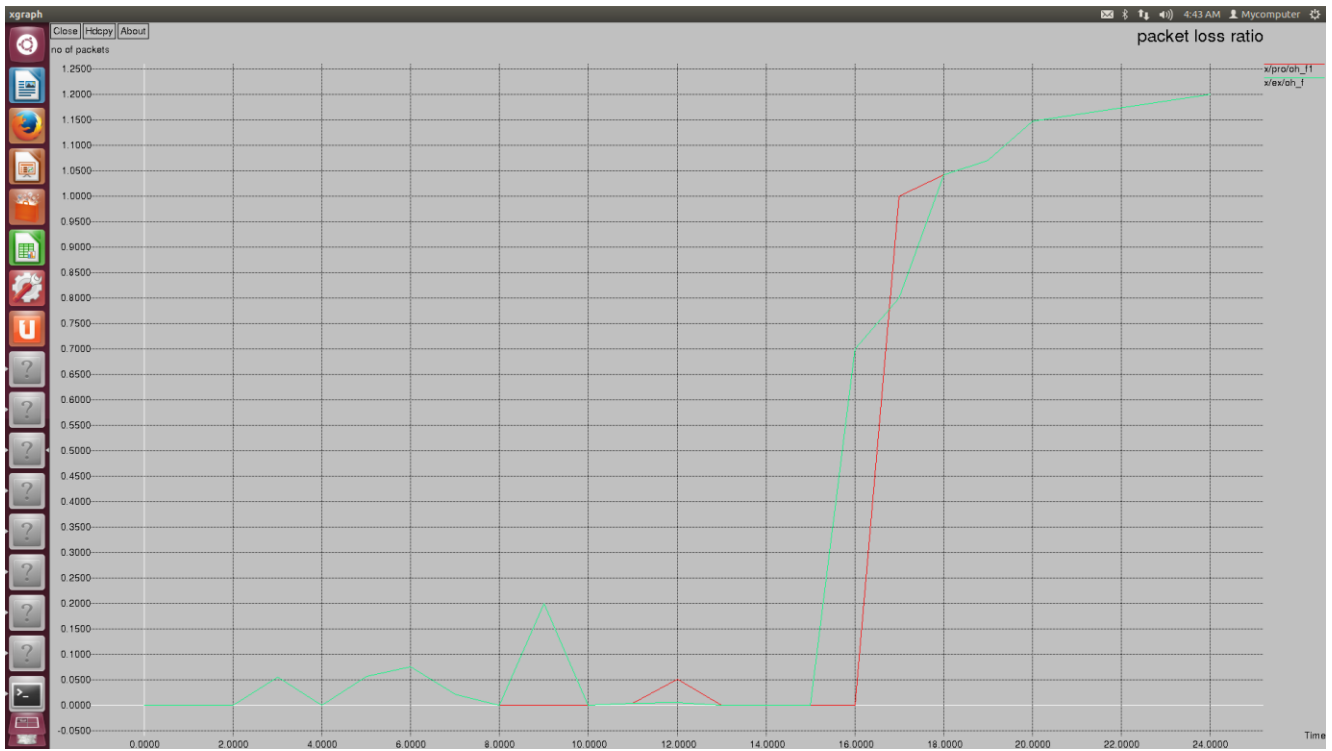


5.9.3 Routing path between Source and Destination

5.6.1 RC4 – Parameter Analysis



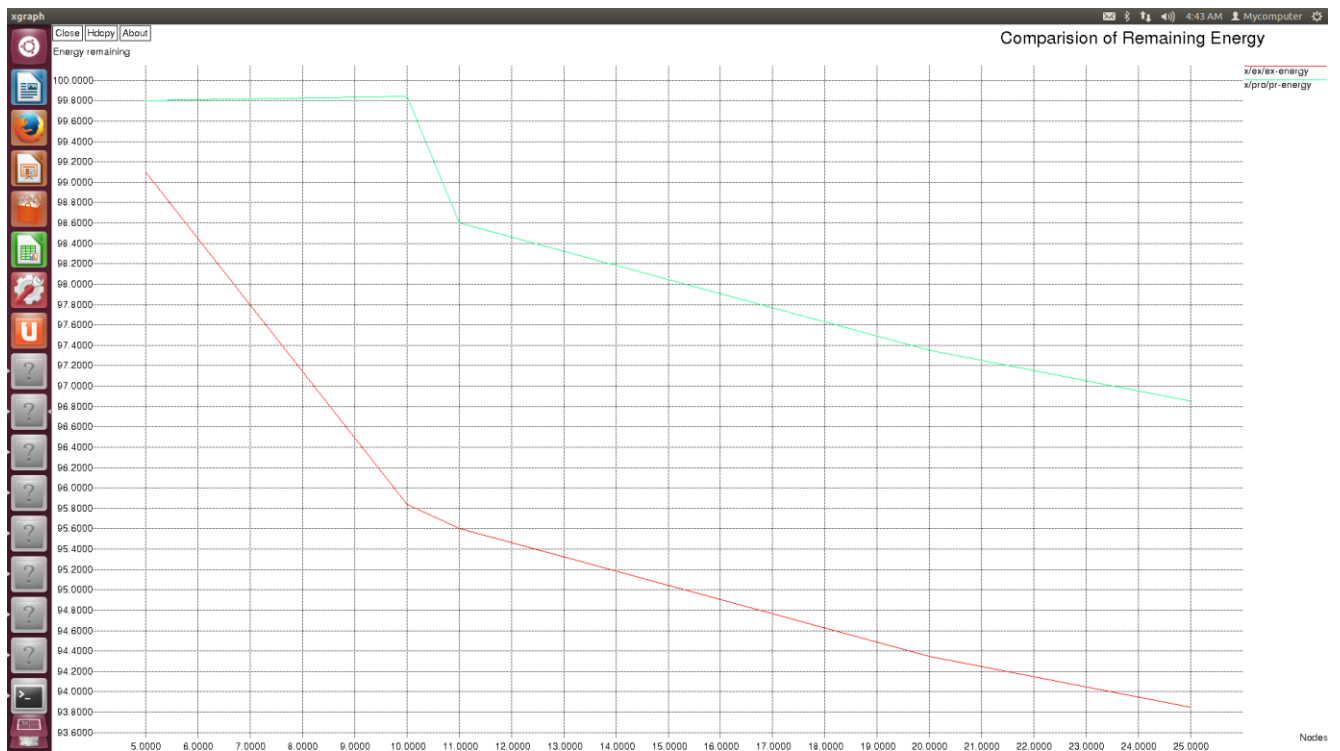
5.10(a) Throughput



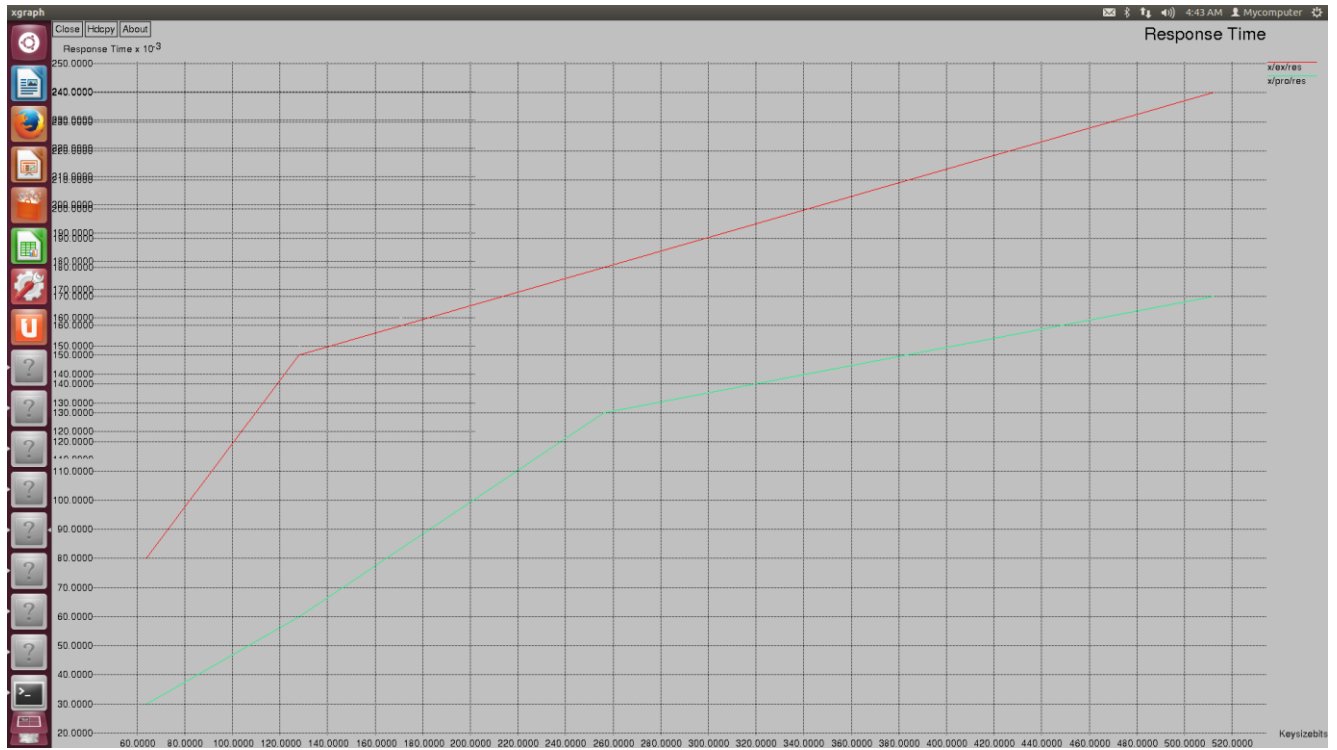
5.10(b) Packet Loss Ratio



5.10(c) Packet Delivery Ratio



5.10(d) Comparison of Energy



5.10(e) Response Time

6.SUMMARY AND CONCLUSION

Getting basic information move about the climate in military applications are essential since such information aid dynamic interaction. Late secure methodologies should be followed for moving the information. The got approaches utilized for moving the top-k question bring about the proposed work help with moving the basic information's safer in the multitier design of WSN. It in this way brings about accomplishing the outcome uprightness and fulfillment confirmation which is fundamental in numerous sensor related applications specifically modern non military personnel zones, territory checking and climate observing.