Volume 3, Issue 10, October 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Formal Verification of Bluetooth Standard 802.11a Using UPPAAL

Er. Gurpreet Singh[1]
Research Scholar,
M-tech, CSE Dept., SBSIET, Padhiana, India

Er. Sandeep Kaur[2]
Assistant Professor,
I T Department, SBSIET, Padhiana, India

Abstract—The time domain and node connectivity is the most critical analytical approach as an analyser in a real time operating system. The purpose of using AODV routing protocol is to decrease the delay and increase the efficiency of the network. AODV routing protocol and employs the idea of a trust model to protect routing behaviours in the network layer of MANETs. Cooperation between nodes should be established without any centralized authority. In the TAODV, trust among nodes is represented by opinion. In secured trust can be used to enhance tight security in the network. Hence UPPAAL is a tool which is specially designed to verify the time constraint in a real time operating system. UPPAAL is appropriate for systems that can be modelled as a collection of non-deterministic processes with finite control structure and real-valued clocks, communicating through channels or shared variables. UPPAAL is a simulator that consists of three parts that are: Editor, Simulator and Verifier. The connectivity of transmitter and receiver is confirmed by verifying the receivers reply to the transmitter and data connectivity is verified by checking the energy level of receiver.

Keywords— Ad- hoc network, TAODV, Routing Protocol, 802.11, UPPAAL.

I. INTRODUCTION

In a distance vector each node knows its neighbours and to reach them. A node maintains its own routing table, storing all nodes in the network, the distance and the next hop to them. If a node is not reachable the distance to it is set to infinity. Every node sends its neighbours periodically its whole routing table. So they can check if there is a useful route to another node using this neighbour as next hop. When a link breaks a Count-To- Infinity could happen. AODV is an 'on demand routing protocol with small delay. That means that routes are only established when needed to reduce traffic overhead. AODV supports three parameters without any further protocols:-

- Unicast
- Broadcast and
- Multicast

A. Unicast Routing

For unicast routing three control messages are used: RREQ (Route REPly), RREP (Route REPly), RERR (Route ERRor).If a node wants to send a packet to a node for which no route is available it broadcasts a RREQ to find one. A RREP includes a unique identifier, the destination IP address and sequence number, the source IP address and sequence number as well as a hop count initialized with zero. If it does not know a route to the destination it rebroadcasts the updated RREQ especially incrementing the hop count. If it knows a route to the destination it creates a RREP. The RREP is unicasted to the origin node taking advantage of the reverse routes. A RREP contains the destination IP address and sequence number, the source IP address, a time to life, a hop count as well as a prefix only used for subnets and some flags. When a node receives a RREP it checks if the hop counts in the RREP for the emitter of the message is lower than the one in its own routing table or the destination sequence number in the message is higher than the one in its own routing table. In MANET link breakage is very common. Routes and link lifetime are extended by sending a package over it and by hello messages. A hello is a special RRER which is only valid for its neighbours. A node may broadcast periodically a hello message so that no link breakages are assumed by its neighbours when they do not hear anything from it for a long time. If a link in an active route breaks a node can try to repair the rout locally. Some other special mechanisms are used like precursors to track the list of active routes for using in RERR emission [1] [2].

B. Multicast Routing

One of the great advantages of AODV is its integrated multicast routing. In a multicast routing table the IP address and the sequence number of the group are stored. To join a multicast group a node has to send an RREQ to the group address with the join flag set. Any node in the multicast tree which receives the RREQ can answer with a RREP. Like this a requester could receive several RREP from which he can choose the one with the shortest distance to the group. A MACT (Multicast Activation) Message is send to the chosen tree node to activate this branch. If a requester does not receive a RREP, the node supposes that there exists no multicast tree for this group in this network segment and it becomes the group leader. A multicast RREP contains additional the IP of the group leader and the hop count to the next group member. The group leader broadcasts periodically a group hello message (a RREP) and increments each time the sequence number of the group. When two networks segments become connected, two partitioned group trees have to be connected.

C. Packet Broadcast in the AODV protocol

The AODV Routing protocol uses an on-demand approach for finding routes, that is, a route is established only when it is required by a source node for transmitting data packets. It employs destination sequence numbers to identify the most recent path. AODV offers quick adaptation to dynamic link conditions, low processing and memory overhead, low memory utilization, and determines unicast routes to destinations within ADHOC network. The message types defined by the AODV protocol are Route Requests (RREQs), Route Replies (RREPs) and Route Errors (RERRs). However, in AODV, the source node and the intermediate nodes store the next-hop information corresponding to each flow for data packet transmission. In an on-demand routing protocol, the source node floods the RREQ packet in the network when a route is not available for the desired destination.

A node offers connectivity information by broadcasting local Hello messages as follows.[7] During every Hello interval milliseconds, the node checks whether it has sent a broadcast within the last Hello Interval.

D. AODV Protocol Overview

The AODV routing protocol is a reactive routing protocol; therefore, routes are determined only when needed. Hello messages may be used to detect and monitor links to neighbours. If Hello messages are used, each active node periodically broadcasts a Hello message that all its neighbours receive. Because nodes periodically send Hello messages, if a node fails to receive several Hello messages from a neighbour, a link break is detected. When a source has data to transmit to an unknown destination, it broadcasts a Route Request (RREQ) for that destination [4]. At each intermediate node, when a RREQ is received a route to the source is created. If the receiving node has not received this RREQ before, is not the destination and does not have a current route to the destination, it rebroadcasts the RREQ. If the receiving node is the destination or has a current route to the destination, it generates a Route Reply (RREP). The RREP is unicast in a hop-by hop fashion to the source. As the RREP propagates, each intermediate node creates a route to the destination. When the source receives the RREP, it records the route to the destination and can begin sending data. If multiple RREPs are received by the source, the route with the shortest hop count is chosen. As data flows from the source to the destination, each node along the route updates the timers associated with the routes to the source and destination, maintaining the routes in the routing table. If a route is not used for some period of time, a node cannot be sure whether the route is still valid; consequently, the node removes the route from its routing table. If data is flowing and a link break is detected, a Route Error (RERR) is sent to the source of the data in a hop-by hop fashion. As the RERR propagates towards the source, each intermediate node invalidates routes to any unreachable destinations. When the source of the data receives the RERR, it invalidates the route and reinitiates route discovery if necessary.

E. AODV WITH BLUETOOTH

Bluetooth is a promising wireless technology that enables portable devices to form short-range wireless network. The basic unit of Bluetooth network, Piconet, can only connect up to 8 nodes. Natural requirement to develop technology that can connect multiple piconets to form a Scatternet while honouring the limitations of the nodes even if the most possible use case scenarios do not predict connectivity among more than 8 devices. Scatternet is an interconnected group of piconets. The node that connects multiple piconets is called PMP (Participant in Multiple Piconet) in Bluetooth specification. There are two types of PMP nodes. The node that attends multiple piconets simultaneously only as slave is called as S/S PMP, and the node that attends multiple piconets simultaneously, and has a master role in one of the piconets, is called M/S PMP in this paper.

The existing ad hoc routing proposals inefficiently use the resources provided by Bluetooth MAC layer. From the experimental results obtained by running AODV on a Bluetooth Scatternet, we noticed that the system performance is greatly influenced by the topology of network, such as the number of slaves connected to a master, and the number of piconets connected to a PMP node. These observations lead to investigate a cross-layer optimization approach to increase the performance of QoS Extended AODV. A word of caution is that we have not used the word optimization in its strict mathematical sense, but to denote the method by which the routing protocol utilizes additional information for improving its performance. In other words, the routing protocol is tuned (optimized) to work with BT specific Scatternets. We present the phenomena that can directly influence the performance of routing protocol over Bluetooth PAN. In section 4, we analyse the reasons for the performance degradation. we first present two cross-layer optimization methods that alleviate the problem of network performance degradation, and then our new routing protocol, Cross layer Optimized Routing for Bluetooth (CORB).

II. IEEE 802.11 STANDARD

The IEEE 802.11 Standard is by far the most widely deployed wireless LAN protocol. This standard specifies the physical, MAC and link layer operation we utilize in our testbed. Multiple physical layer encoding schemes are defined, each with a different data rate. FHSS and DSSS techniques are used for spreading signal in different data rate.

1) IEEE 802.11a: In terms of speed, the 802.11a standard was far ahead of the original 802.11 standards. 802.11a specified speeds of up to 54Mbps in the 5GHz band, but most commonly, communication takes place at 6Mbps, 12Mbps, or 24Mbps. 802.11a is incompatible with the 802.11b and 802.11g wireless standards.
2) IEEE 802.11b: The 802.11b standard provides for a maximum transmission speed of 11Mbps. However, devices are designed to be backward-compatible with previous 802.11 standards that provided for speeds of 1, 2, and 5.5Mbps. 802.11b uses a 2.4GHz RF range and is compatible with 802.11g.
3) IEEE 802.11g: 802.11g is a popular wireless standard today. 802.11g offers wireless transmission over distances of 150 feet and speeds up to 54Mbps compared with the 11Mbps of the 802.11b standard. Like 802.11b, 802.11g operates in the 2.4GHz range and therefore is compatible with it.

III. Implementation Design

The AODV routing daemon to function it must determine when to trigger AODV protocol events. Since the IP stack was designed for static networks where link disconnections are infrequent and packet losses are unreported, most of these triggers are not readily available. Therefore, these events must be extrapolated and communicated to the routing daemon via other means. The events that must be determined are:

1) *When to initiate a route request:* This is indicated by a locally generated packet that needs to be sent to a destination for which a valid route is not known.
2) *When and how to buffer packets during route discovery:* During route discovery packets destined for the unknown destination should be queued. If a route is found the packets are be sent.
3) *When to update the lifetime of an active route:* This is indicated by a packet being received from, sent to or forwarded to a known destination.
4) *When to generate a RERR if a valid route does not exist:* If a data packet is received from another host and there is no known route to the destination, the node must send a RERR so that the previous hops and the source halt transmitting data packets along this invalid route.
5) *When to generate a RERR during daemon restart:* After the AODV routing protocol restarts, it must send a RERR message to other nodes attempting to use it as a router. This behaviour is required in order to ensure no routing loops occur. In the remainder of this section we discuss various design approaches. First, we examine how to determine these events and where to place the AODV protocol logic. We describe the advantages and disadvantages of each solution, and we justify why we chose a user-space daemon with a small kernel module. In addition, we discuss the importance of monitoring neighbour connectivity and how it is performed.

A. Formal verification

In order to increase the quality of the protocols, designers of the protocol must validate the design of the protocol. There are several methods for validation like using a software model of protocol and performing it on virtual devices in simulated environment or doing live test on real hardware. These tests can be used to identify bugs but cannot explode protocol blue print error. Formal verification is the technique used to authenticate protocols.

Formal verification is a systematic process that uses mathematical reasoning for the specification, development, and verification of computer–based software and hardware systems [6]. It provides the systematic way to access the correctness of protocol, process, and system. It checks whether a design satisfies some requirements (properties). There are three types of formal verification techniques.

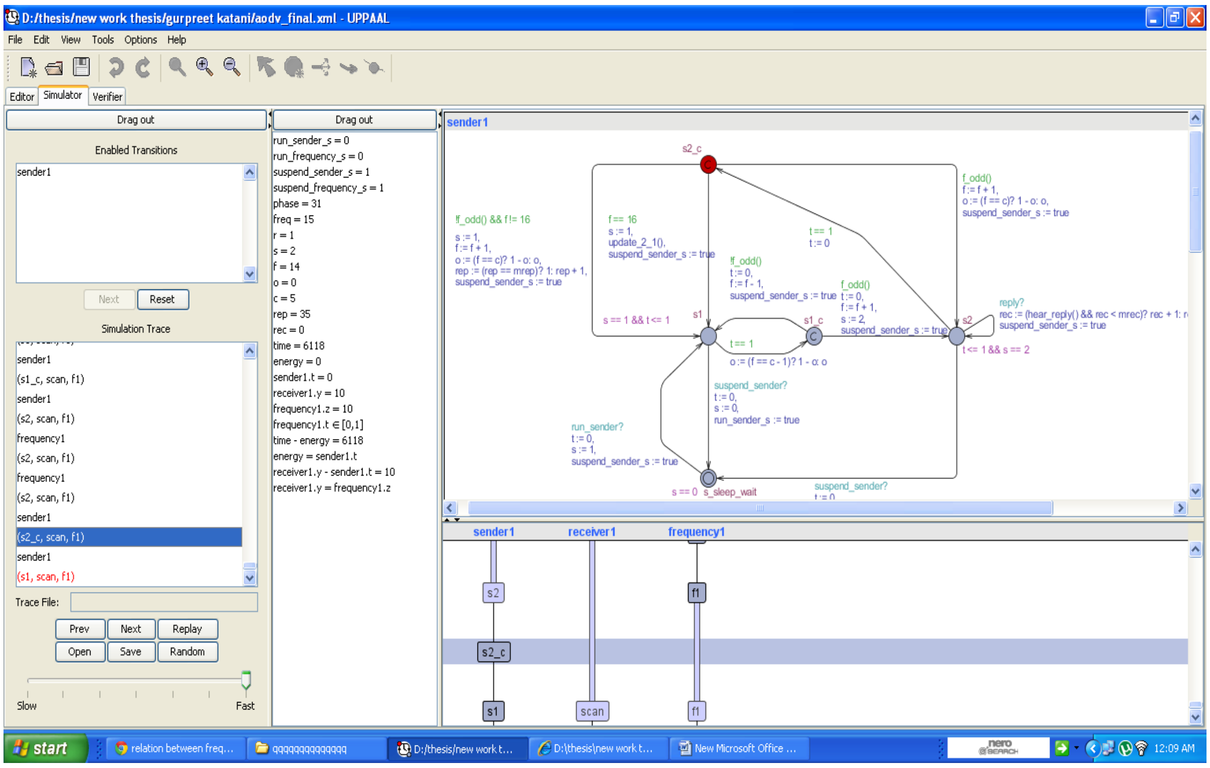
- Model checking: It is the technique in which a system verifies certain properties by means of an exhaustive search of all possible states that a system could enter during its execution.
- Theorem proving: It is the technique in which a system attempts to produce a formal proof, given a description of the system, a set of logical axioms, and a set of inference rules.
- Equivalence checking: It formally checks the equivalence of two models, which are at different abstraction level.

Model checking is a method to verify whether a formally modeled system satisfies a given logic specification [3]. In automated model checking the prove tool often varies some properties using an exhaustive search of all possible states that the model could reach during its execution. Model Checking and Theorem Proving techniques are used for the validation of routing protocols of ad hoc networks. There are various tools and techniques used for formal modeling and verification of the ad hoc network routing protocols. It ranges from Petri nets, SPIN Model Checker, PROMELA, AVISPA, HLPSL, UPPAAL Model Checker, SDL and BAN logic.

IV. Uppaal

UPPAAL is a verification tool for timed automata. Its focus on speed and usability has made the tool popular both as a teaching tool in academia, as a gentle introduction to the world of model checking, and as a tool for doing serious case studies as witnessed by the large number of publications in which UPPAAL was used. Version 4.0 was released in May 2006 and introduces many new features that increase the applicability and the performance of the tool. This paper describes three major features of the new release. Many more are worth describing (such as a typical reduction in memory usage of a factor 3 to 5, new abstraction techniques resulting in large performance increases [5], or our implementation of the generalised sweep line method), however lack of space forces us to focus on the most visible changes. UPPAAL uses client-server architecture and split the tool into a graphical user interface and a model-checking engine. The user interface, or client, is implemented in Java and the engine, or server, is compiled for different platforms (Linux, Windows, Solaris). The GUI interface of UPPAAL has three main tool components: the system editor, the simulator, and the verifier. These tools are integrated on one common interface and work on same internal system model. This makes the exchange of information easier, e.g. loading diagnostic trace generated by the verifier into the simulator for further inspection.[11]

V. SIMULATION RESULT AND ANALYSIS


D:/thesis/new work thesis/gurpreet katani/aodv_final.xml - UPPAAL

Enabled Transitions

sender1

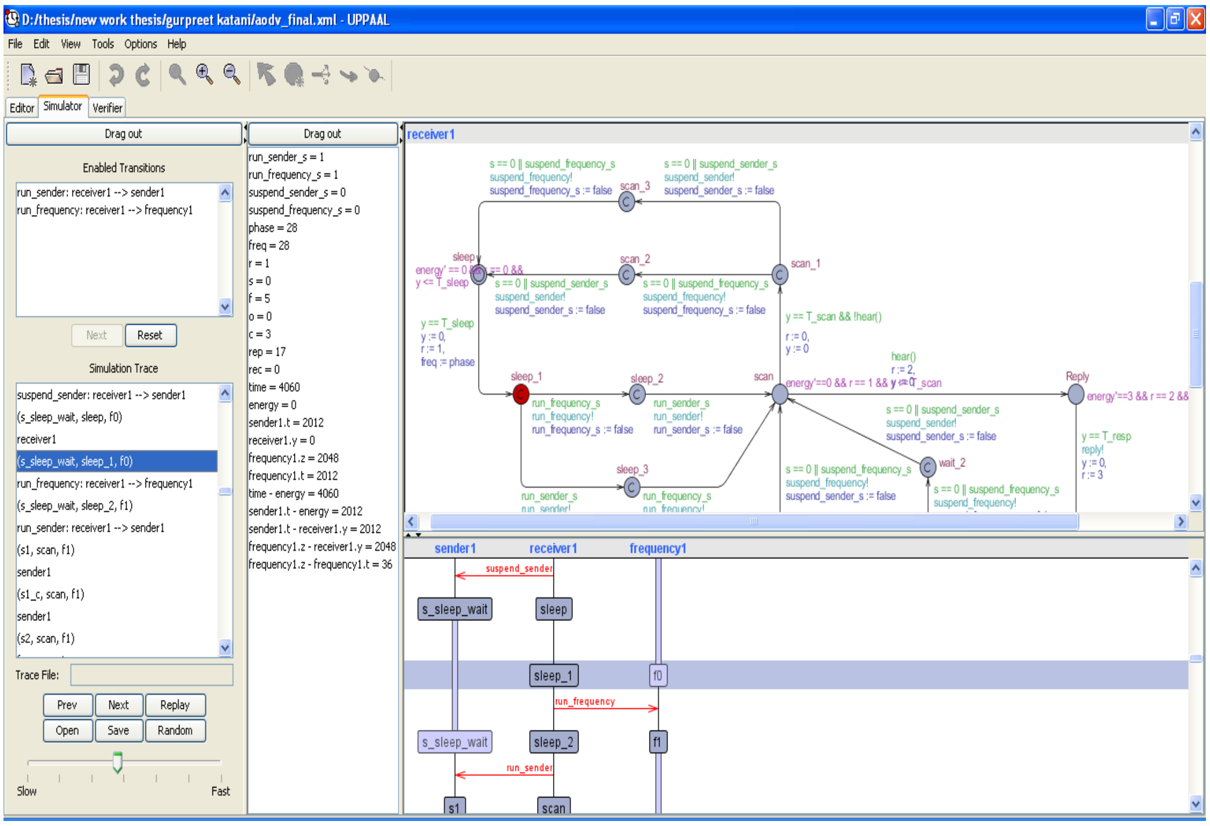
Simulation Trace

```
sender1
(s1_c, scan, f1)
sender1
(s2, scan, f1)
frequency1
(s2, scan, f1)
frequency1
(s2, scan, f1)
sender1
(s2_c, scan, f1)
sender1
(s1, scan, f1)
```

```
run_sender_s = 0
run_frequency_s = 0
suspend_sender_s = 1
suspend_frequency_s = 1
phase = 31
freq = 15
r = 1
s = 2
f = 14
o = 0
c = 5
rep = 35
rec = 0
time = 6118
energy = 0
sender1.t = 0
receiver1.y = 10
frequency1.z = 10
frequency1.t ∈ [0,1]
time - energy = 6118
energy = sender1.t
receiver1.y - sender1.t = 10
receiver1.y = frequency1.z
```

Fig.1 simulation of sender

In sender s1 and s2 are the main nodes for transferring and acknowledging the data. For condition s1_c and s2_c are used. While acknowledging s2_c follow these conditions: if s=2, rep=rep+1, suspend_sender_s= true. When data reach at the s1 , rep incremented by 1.


D:/thesis/new work thesis/gurpreet katani/aodv_final.xml - UPPAAL

Enabled Transitions

```
run_sender: receiver1 --> sender1
run_frequency: receiver1 --> frequency1
```

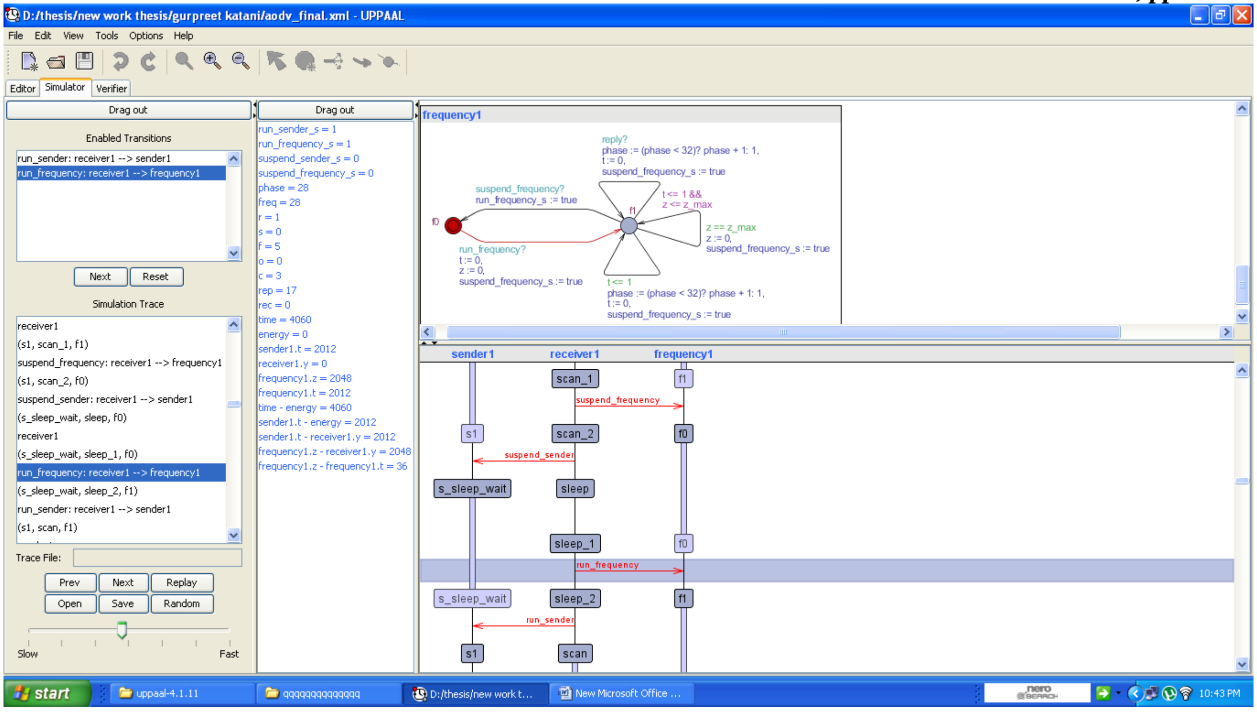
Simulation Trace

```
suspend_sender: receiver1 --> sender1
(s_sleep_wait, sleep, f0)
receiver1
(s_sleep_wait, sleep_1, f0)
run_frequency: receiver1 --> frequency1
(s_sleep_wait, sleep_2, f1)
run_sender: receiver1 --> sender1
(s1, scan, f1)
sender1
(s1_c, scan, f1)
sender1
(s2, scan, f1)
```

```
run_sender_s = 1
run_frequency_s = 1
suspend_sender_s = 0
suspend_frequency_s = 0
phase = 28
freq = 28
r = 1
s = 0
f = 5
o = 0
c = 3
rep = 17
rec = 0
time = 4060
energy = 0
sender1.t = 2012
receiver1.y = 0
frequency1.z = 2048
frequency1.t = 2012
time - energy = 4060
sender1.t - energy = 2012
sender1.t - receiver1.y = 2012
frequency1.z - receiver1.y = 2048
frequency1.z - frequency1.t = 36
```

Fig.2 simulation of receiver

In this figure the data starts from initialized node to highlighted node, when the condition y=0, r=1, freq= phase, frequency1.z-frequency1.t = 36 will be true. When this condition occurs then phase value will becomes frequency value.


D:/thesis/new work thesis/gurpreet katani/aodv_final.xml - UPPAAL

Enabled Transitions

run_sender: receiver1 --> sender1
run_frequency: receiver1 --> frequency1

Simulation Trace

receiver1
(s1, scan_1, f1)
suspend_frequency: receiver1 --> frequency1
(s1, scan_2, f0)
suspend_sender: receiver1 --> sender1
(s_sleep_wait, sleep, f0)
receiver1
(s_sleep_wait, sleep_1, f0)
run_frequency: receiver1 --> frequency1
(s_sleep_wait, sleep_2, f1)
run_sender: receiver1 --> sender1
(s1, scan, f1)

```
run_sender_s = 1
run_frequency_s = 1
suspend_sender_s = 0
suspend_frequency_s = 0
phase = 28
freq = 28
r = 1
s = 0
f = 5
o = 0
c = 3
rep = 17
rec = 0
time = 4060
energy = 0
sender1.t = 2012
receiver1.y = 0
frequency1.z = 2048
frequency1.t = 2012
time - energy = 4060
sender1.t - energy = 2012
sender1.t - receiver1.y = 2012
frequency1.z - receiver1.y = 2048
frequency1.z - frequency1.t = 36
```

Fig.3 simulation of frequency

In fig. 3 f0 and f1 frequencies are used for hopping.f1 suspended their signal and comes in f0 state. This situation occurs when run_frequency_s = true. When the sender and receiver come to their original state to send and acknowledge data then frequency again go to f1 state. If the condition is suspended_frequency_s = true.

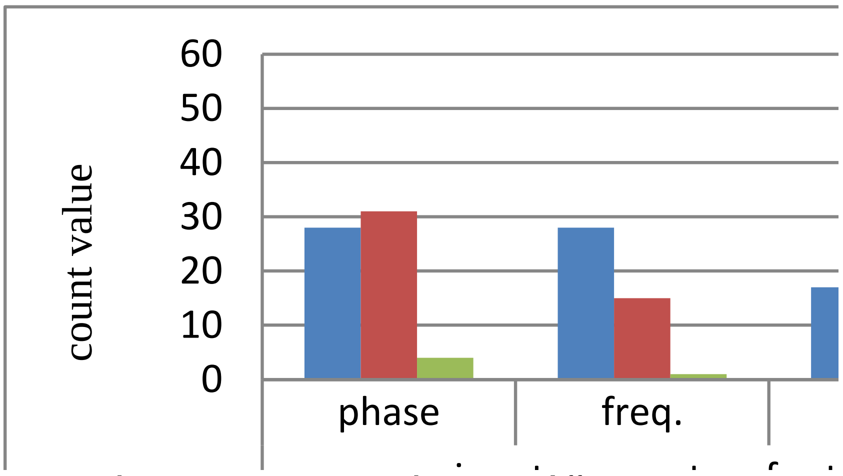


Fig.4 showing time constraints with respect to different parameters

The transmitter and receiver verified the properties in the tool. These properties are for checking the connectivity of transmitter and receiver and reception of data among them.

VI. Conclusions

We have modelled the AODV protocol with timed automata and analyzed certain configurations using a number of techniques developed for the UPPAAL model checker to obtain model reductions. We observe that the protocol as specified may unnecessary result in failure to discover the route or deliver the message the problem occurs because nodes wait for a fixed time that in a dynamically growing network may take much longer to reach the requesting node. We propose a modification to the AODV routing algorithm by allowing the nodes to amend the value of Path Followed and NET_DIAMETER through learning about the size of the network from the incoming packets and time analysis to our knowledge this is the first solution to this problem. The performance of AODV can be further enhanced using fuzzy logic by taking different input parameters to reduce the uncertainty for finding an optimal path.

References

[1] A. Verma, “Formal Verification of Ad hoc Networks Routing Protocols”, International Journal of Advanced Research in Computer Science, Volume 2, No.4, July-August 2011

[2] S. Sharma, “A Review on Different Routing Protocols in Ad-Hoc Network”, An International online open access peer reviewed journal 2012, pp. 53-60

[3] A. David, J. Hakansson, K. G. Larsen, and P. Pettersson, “Model checking timed automata with priorities using dbm subtraction,” 2006, submitted for publication.

[4] O. Wibling, J. Parrow, and A. N. Pears, “Automatized verification of ad hoc routing protocols,” in Formal Techniques for Networked and Distributed Systems – FORTE, ser. LNCS, vol. 3235. Springer, 2004, pp. 343–358.

[5] G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelnek, “Lower and upper bounds in zone-based abstractions of timed automata,” International Journal on Software Tools for Technology Transfer, September 2005 (2002) The IEEE website. [Online]. Available: http://www.ieee.org/

[6] D. L. Dill, “Timing assumptions and verification of finite-state concurrent systems.” in Proc. Of Automatic Verification Methods for Finite State Systems, ser. LNCS, vol. 407. Springer–Verlag, 1989, pp. 197–212/

[7] C. Perkins, E. Belding-Royer, and S. Das, “Ad hoc on-demand distance vector (AODV) routing,” RFC 3561, 2003, http://www.ietf.org/

[8] R. Perlman, “An algorithm for distributed computation of a spanningtree in an extended lan,” SIGCOMM Comput. Commun. Rev., vol. 15, no. 4 pp. 44–53, 1985.

[9] M. Hendriks, “Model checking the time to reach agreement,” in 3rd International Conference on the Formal Modeling and Analysis of Timed Systems (FORMATS’05), ser. LNCS, P. Pettersson and W. Yi, Eds., no. 3829. Springer–Verlag, 2005, pp. 98–111.

[10] [18] B. Gebremichael, F. Vaandrager, and M. Zhang, “Analysis of a protocol for dynamic configuration of IPv4 link local addresses using Uppaal,” ICIS, Radboud University Nijmegen, Tech. Rep. ICIS-R06xxx, 2006, submitted to EMSOFT 2006.

[11] M. Stigge, “UPPAAL 4.0: Small Tutorial”, Available at: http://www.it.uu.se/research/group/darts/uppaal/small_tutorial.pdf, Nov 16, 2009.

[12] S. M. Schwartz, “Frequency Hopping Spread Spectrum (FHSS) vs. Direct Sequence Spread Spectrum (DSSS) in Broadband Wireless Access (BWA) and Wireless LAN (WLAN)”, Available at: http://sorin-schwartz.com/white_papers/fhvsds.pdf.