State Transition Testing Approach for Ad hoc Networks using Ant Colony Optimization

Ahmed Redha Mahlous, Anis Zarrad, Taghreed Alotaibi
Computer Science Department, Prince Sultan University
PSU, Riyadh, Saudi Arabia

***Abstract*—Nowadays, telecommunication software organizations are challenged to provide high-quality software to customers within their estimated time and budget in order to stay competitive within the market. Because quality is a defining aspect of the product, it is essential for a project manager to stay alert throughout the project lifecycle. Quality has a direct bearing on customer satisfaction, and if a company produces high-quality products, satisfied customers will rank it highly in customer satisfaction surveys. Additionally, dissatisfied customers are more vocal in their criticisms. Therefore, testing is an important step to produce more reliable systems. In this paper we address two important aspects of software testing for ad hoc network protocols. The first one is by integrating a high-level testing approach based on state transition on top of a network simulator in order to fill a perceived gap in existing network simulators. The second one is reducing testing effort by eliminating redundant test cases, in order to effectively improve the result accuracy of existing network simulators. In this paper, we implemented an automated state transition testing approach for wireless network routing protocols, using an improved Ant Colony Optimization (ACO) algorithm. The expected result is to provide maximum coverage in terms of states and transitions.**

Keywords—Component; ant colony; simulation; optimization; state transition; ad hoc routing protocol

I. Introduction

In the last few decades, competition in the software market has increased, and software developers are working on high quality products with limited time and budget. Quality is essential in every phase of the project lifecycle as it has a direct bearing on customer satisfaction. Customer dissatisfaction is harmful to a company’s reputation, which is why the testing phase is crucial for developing high-quality, reliable systems.

In the testing phase, developers focus on investigation and discovery, finding out whether their software works in line with customer requirements. Using the results from this phase makes it possible to reduce the number of errors within the software program because it's not possible to solve all the failures you might find during the testing phase [1].

Typically, network protocols are modeled using state-machine diagrams [2] that consist of a finite number of protocol states with or without connections between them. Connections between states are called events, and they facilitate the transition from one state to another. Network protocols are systems with large input and output parameters. For this reason it is necessary to find testing solutions that reduce parameter problems such as duplicates in the path, and at the same time increase the overall effectiveness of the testing. Our method of achieving this was by introducing a state transition testing approach, which is placed on the top of the network simulator in order to ensure an effective and optimal number of test cases.

Due to the complexity of networks, simulators play an important role in overcoming some of challenges that arise from implementing and testing network protocols. However, simulators also have their limitations [3], e.g. traffic generation, documentation, and scalability, especially with the current growth in the use of wireless devices. Furthermore, simulators do not provide an accurate portrayal of real life because they use the queuing theory and discrete events. For example if network congestion is high, estimation of the average occupancy becomes challenging due to high variance [3]. To this end, there is a need to integrate a new software testing approach in order to overcome potential weaknesses in the simulation environment.

Testing is an important phase in the software development lifecycle. The developed software needs to be tested thoroughly to ensure that it meets the needs and expectations of its users, and that it is free from errors. Before starting with the actual validation, testing activities must be planned properly in order to perform effective testing. Delivering a high quality product is crucial for maintaining customer satisfaction and reducing the risk of faults and the cost of repairing them. Quality contributes to the long-term revenue and profitability for companies, making it possible for them to charge and maintain higher prices for their products. Our main focus in this research was to integrate a high-level testing approach into a network simulator to guarantee effective testing. Traditionally, network research communities implement their proposed solutions in a network simulator and generate scenarios based on the network scale, node mobility, and traffic generation models. This approach can limit the scenario size and traffic model exceptions in the source code. Our focus is on the functionalities of network protocols at every point of the simulation.

This research adopts a quantitative methodology. First, related works from journals and conference proceedings will be reviewed. Then, data collection and statistical analysis of Ant Colony Optimization (ACO) will be performed for ad hoc networks. Finally, the proposed approach will be implemented and compared with existing approaches. Fig. 1 illustrates the stages of the methodology.

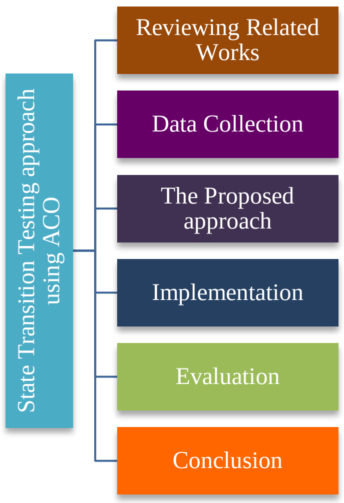


Fig. 1. Methodology.

This paper is organized as follows: Section II provides background and related works; Section III presents the proposed approach, Section IV presents the improvements in ACO and implementation details. Section V describes the results. Section VI provides a comparison between the original ACO and the improved one; and finally Section VII concludes the paper.

II. Background and Related Works

A. Background

In this section, we present a brief description of the protocols and algorithm used. Two protocols have been chosen and considered as a good choice for MANETS [6], DSDV and AODV.

1) *Ad hoc On-Demand Distance Vector (AODV)*

AODV is a routing protocol for ad hoc mobile networks with a large number of mobile nodes. The protocol's algorithm creates routes between nodes only when the source nodes request the routes. This protocol allows the network to be more flexible in order to give nodes the choice to enter and leave the network. The routes will be active unless there are no data packets being sent from the source to the destination. Once the source stops sending packets, the path will be time out and close. AODV additionally supports both unicast and multicast.

The protocol has different types of messages. It initiates the request when needed (“on demand”). Then, route discovery starts with “route request” and “route reply” messages. Finally, routes are maintained just as long as necessary. There are four types of messages used for communication among the nodes. Route Request (RREQ) and Route Reply (RREP) messages are used for route discovery. RREQ packets are broadcast by the source node to connect with the destination node, as the source node has no route entry to the destination node. For the RREP, it unicasts the message to the source node if the node is the destination or has a route to the destination. The other two types are Route Error (RERR) messages and Hello messages, which are used for route maintenance [7].

2) *Destination-Sequenced Distance-Vector Routing (DSDV)*

DSDV is a proactive routing protocol and a table-driven routing scheme for ad hoc mobile networks requiring each node to periodically broadcast routing updates. This is a table-driven algorithm based on modifications made to the Bellman-Ford routing mechanism. Each node in the network maintains a routing table that has entries for each of the destinations in the network and the number of hops required to reach each of them. Each entry has a sequence number associated with it that helps in identifying stale entries.

Each node periodically sends updates tagged throughout the network with a monotonically increasing even sequence number to advertise its location. New route broadcasts contain the address of the destination, the number of hops to reach the destination, the sequence number of the information received regarding the destination, as well as a new sequence number unique to the broadcast. The route labeled with the most recent sequence number is always used. When the neighbors of the transmitting node receive this update, they recognize that they are one hop away from the source node and include this information in their distance vectors. Every node stores the “next routing hop” for every reachable destination in their routing table. The route used is the one with the highest sequence number, i.e. the most recent one. When a neighbor B of A finds out that A is no longer reachable, it advertises the route to A with an infinite metric and a sequence number one greater than the latest sequence number for the route forcing any nodes with B on the path to A to reset their routing tables [8].

3) *State transition testing (STT)*

State Transition testing is a testing technique in which changes in input conditions causes state changes in the application under test. It is a process where the tester analyses the behaviour of an application under test for different input conditions in a sequence. The tester provides both positive and negative input test values and records the system’s behaviour. It is helpful where testing different system transitions is needed. The state transition testing model consists of four parts: states, transitions, events, and actions [9].

Moreover, state transition has state diagram, events and test cases as its output, and it is commonly used in black box testing. It uses model of the states for the component to occupy the transitions between those states, the events which cause those transitions, and the actions which may result from those transitions [9].

a) Ant colony optimization (ACO)

Ant Colony Optimization (ACO) is a metaheuristic for solving computational problems. Biologists noticed that blind ants can find the shortest path between food sources and their nests. They also found out that ants spread pheromones on their way, and so other ants can follow the previous pheromone trail by while traversing the paths. The higher the amount of pheromone present, the higher the probability that the ants will follow the trail. This indirect communication between the ants via pheromone trails allows them to find the shortest paths between their nests and food sources [10].

Metaheuristic algorithms are used to escape from local optima, control some basic heuristic: by constructing a heuristic starting from a null solution and adding elements to build a complete one, or a local search heuristic starting from a complete solution and iteratively modifying some of its elements in order to achieve a better one. The metaheuristic part permits the low-level heuristic to obtain solutions better than those it could have achieved alone, even if iterated [10].

B. Related Works

In this section, previous works related to the use of ant colony optimization for testing wireless networks are presented. We also present some of the most used network simulators by the network research community.

Authors in [11] proposed a formal model named "Evolving-Graph-Based Finite State Machine" (EGFSM) to describe the behaviors of protocols in a mobile ad hoc network for conformance testing. To enhance the description capacity for the dynamic behavior, the authors proposed a method to introduce the evolving graph theory to extend the Finite State Machine (FSM) model. They assumed that the topology of the protocol under testing is predictable. The test sequences generated from the proposed model can be adapted in test execution for specific network topologies. Finally, they presented a case study to validate the effectiveness of the proposed model and its generated test sequences.

The authors of [12] proposed a framework for testing wireless sensor networks and presented testing strategies for the same. Their paper addressed some of the characteristics of WSNs, such as node dependencies, the location of the mote, the lack of human interaction during runtime, and how these affect the testing process. The paper mentions two types of testing which must be applied on such applications. The first type is unit testing and it is applied on the motes of the networks. The second one is integration testing and it tests the network after the integration. This type of testing is not applicable for routing protocols since it is based on transitions and searching for paths, and needs intelligent algorithms to automate the testing process.

Authors in [4] proposed an ant colony optimization algorithm to generate an automatic state transition test sequence to offer a strong level of software coverage. The paper focused on providing full coverage and they mentioned that there is a previous work [5] that has applied ACO but with less coverage, while other papers used genetic algorithms to improve the quality of the testing, and neither of them considered full software coverage. They also used the visited status concept to ensure that each vertex is visited at least one. This concept is based on the value of the visited status parameter, for example if there is a connection between two vertices and the second one was not visited by any ants before then the ant will select this vertex. They used a tool called STTACO, a genetic algorithm, and their proposed approach, which based on ACO to apply it on real time case studies "enrolment system". The proposed solution can be applied on wireless networks since it is a state machine, and the most important thing is that they provided full coverage.

The authors in [1] proposed ACO to generate a set of optimal paths in the control flow graph (CFG) and prioritize the paths. They also proposed an approach to generate a test data sequence within the domain and use it as the input of the generated paths. Their approach guarantees full software coverage with minimum redundancy, and their proposed algorithm prioritizes paths in two ways to decide which paths are to be tested first. The first way takes a CFG as an input and generates optimal and prioritized paths. The second way uses ACO for test data generation and uses it as the input of the generated paths. In the end, they applied it to a binary search program to generate paths as well as inputs. The benefits of their work are: providing full path coverage through the CFG (node, branch, loop, condition), using ACO to generate paths as well as test data, prioritizing the paths to ensure effective testing, and removing redundancy. The authors used a control flow graph as their input, whereas in this research, a state transition table is used as input.

Authors in [13] described a state machine testing approach for cyber-attacks and malicious activities. A map component is implemented between the system-under-test and the learner component. The learner operates abstract inputs and outputs while the mapper transforms the abstract inputs into concrete ones which are accepted by the system. In the opposite direction, it transforms the concrete outputs into abstract ones for the learner. Simulation results show improved outputs from a security viewpoint.

Authors in [24] presented the method of inference and analysis of formal models of botnet command and control (C&C) protocols. Their contribution was to establish a novel state-machine approach for reducing the number of inputs in a realistic high-latency network environment. A Mealy machine modeling approach [2] was used. Optimization of the L* algorithm was implemented to made the inferring procedure more effective.

Although simulators have a major role in evaluating routing protocols, they also have their limitations. Starting with NS2, which is the first developed simulator, the simulator has some issues related to interoperability and coupling between models. There is also a lack of memory management, debugging of split language objects. NS3 was developed to overcome the problems with NS2, but there is still a need for improvements to the animator tool for wireless scenarios, user friendless, ease of use, as well as good tutorials and wider community support.

There are other simulators such as OPNET [14] and QuaNet [15], but they are not useful for students and researchers because they are commercial network simulators and there are no educational licenses for them. Another limitation related to OPNET is that because it is proprietary software, customization options are limited.

Authors in [16] presented a performance comparison between NS2, NS3, OMNeT++, and GloMoSiM using the AODV routing protocol. They stated that in order to evaluate routing protocols using simulators, three variables must be considered: memory usage, computation time, and CPU utilization [16]. Table I illustrates their comparison.

A similar work was presented in [17] where the authors compared various network simulators such as OPNET [14], NS2 [18], NS3 [19] QualNet [16], OMNeT++ [20], J-Sim, and Backplane, and tested their suitability when used for simulation of critical infrastructure. Other research works [21] and [22] have presented comparative study to compare various simulators. However, they do not give a comparative study, and instead only provide a description of each simulator independently. On the other hand, many researches have been conducted on testing routing protocols using high-order testing approaches such conformance testing [23] and black box testing [24]. Authors in [24] proposed a framework for WSN testing to apply distributed unit testing concepts in the development process.

To the best of our knowledge, no previous work has been done to integrate high order testing techniques on top of a simulator environment for correctness and test case selection. Random test cases can easily hide protocol defects because there are no specific procedures to be followed in the testing stage. Thus, random test cases even though will not cover all the cases. In practice, network simulator software is commonly used in testing and protocol assessment. The trustworthiness of results produced from simulation models must be investigated because simulators are less accurate compared to real-life scenarios, as they use queuing theory and discrete events. Simulator output results are often taken as evidence without further verification. In this work, a new layout-level verification tool is added in order to find critical test cases and test implemented protocols.

TABLE I. PERFORMANCE COMPARISON

	NS2	NS3	OMNeT++	GloMoSiM
Memory usage	Highest amount of memory	Lowest amount of memory	Average	Average
CPU usage	Higher	Higher	Lowest	Lowest
Computation time	Highest computation time	Lowest	Low	Low

III. PROPSOED APPROACH

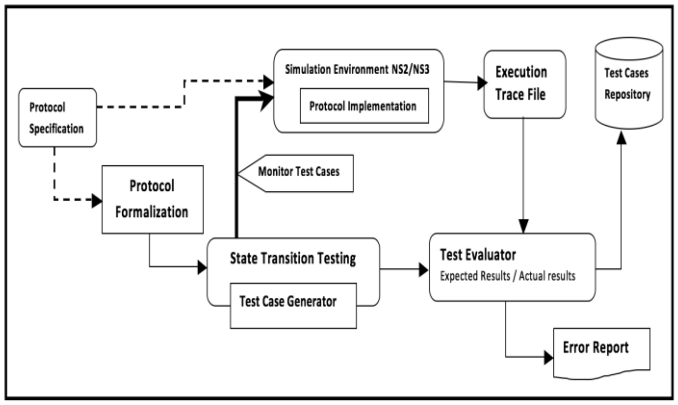


Fig. 2. The proposed approach.

The proposed approach enables testers to be exposed to the design of the protocol and reduces the cost investment.

As shown in Fig. 2, the state transition testing approach was placed on top of the network simulator. The reasons for such decision are to ensure effective test cases along with an optimal number of test cases.

The main components are:

a) Simulation environment: refers the simulation software. The proposed approach can be adapted to any environment, such as NS2, NS3, OMNeT++, etc. In this work, NS2 is used. The trace file entity refers to the actual output generated from NS2. It captures events occurring in the modelled network. The trace data is in ASCII code and is organized into 12 fields.

b) Monitor: is in charge of injecting test cases generated by the state transition testing in the simulation environment. A trace file is generated and passes through the test evaluator module.

c) Protocol Formalization: implementing the state machine formalization model for a specific routing protocol. **Protocol Specification**: represents the protocol's task with a comprehensive description of the intended purpose.

d) State Transition Testing (STT): is the main component for learning about the protocol being tested. This learning is important to create test cases for the simulation environment module. The testing action has two roles: (1) running state transition testing to achieve the highest coverage with minimum redundancy, and (2) transferring generated test cases to the simulation environment module. A complex role is assigned to STT to define test cases based on system knowledge and communication scheme. This is necessary to ensure effective testing with reduced cost. An improvement of the ACO algorithm is implemented in section 4 to enhance the test case quality section and the system performance.

e) Test Evaluator: is used in deciding whether to pass or fail a test case based on protocol formalization and actual/expected results. The expected results are the conditional criteria that show the output that should be generated from the test case. In our case, the expected results when all states and transitions are covered. The test passes if the actual result matches the expected result based on the following specifications:

- All states have been reached at least once.
- All transitions have been executed at least once.
- All feasible paths have been executed.

The actual results are retrieved from the trace file that represents the behaviour observed when a protocol is tested. All tests with a 'pass' are stored in the test case repository for future use in regression testing. Tests with a 'fail' need additional investigation.

A. Apply STT for AODV and DSDV

One of the components in our proposed approach is protocol formalization, which is implementing the state machine formalization model for a specific routing protocol. In this work, state transition testing is applied on wireless network routing protocols AODV and DSDV. Fig. 3 illustrates the AODV protocol using a state diagram. The protocol messages are illustrated as states and transitions. AODV first starts by initiating the request on demand. Then, route discovery is initiated and once it has found one it will request a route and get route reply if a valid route found. After route maintenance, two different types of messages will be sent: Hello and Error.

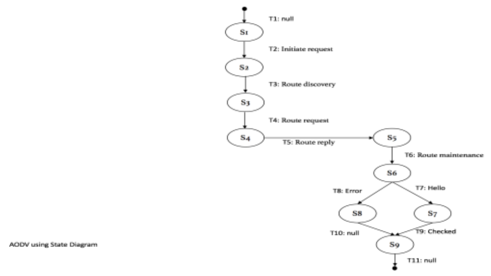


Fig. 3. AODV using STT.

Table II presents the states and events of AODV, which happened in order to move from one state to the other. Each event is described in Table III.

For instance to move from S1 to S2, Event X which corresponds to Initiate Request as shown in Table III should happen first. The same thing applies to the transition from one state to another for all states. Each transition should be done once the corresponding event has occurred.

TABLE II. STATE TRANSITION TABLE FOR AODV

	S 1	S2	S3	S4	S5	S6	S7	S8	S9
S 1		Even t X							
S 2			Even t Y						
S 3				Even t J					
S 4					Even t I				
S 5						Even t K			
S 6							Even t Z	Even t F	
S 7									Even t C
S 8									Even t G
S 9									

TABLE III. DESCRIPTION OF AODV EVENTS

Event	Description
Event X	Initiate Request
Event Y	Route Discovery
Event J	Route Request
Event I	Route Reply
Event K	Route Maintenance
Event Z	Hello
Event F	Error
Event C	Checked
Event G	Null

State transition testing was also applied on DSDV as shown in Fig. 4. The protocol messages are illustrated as states and transitions. It first starts by creating a routing table for each node and monitoring the tables for two types of events. The first type occurs when the update is triggered and the second type is a periodic update. For the triggered update, this means that a new change has been detected. This first type of update packet is sent for a major change, and contains all the routing information available at a node. In this case the required action is applying a full dump. The second type is sent for a minor change in the routing table, e.g. new nodes added or link breakage. This type of update packet contains only the information that has changed since the last full dump was sent out by the node. So, the action is applying incremental.

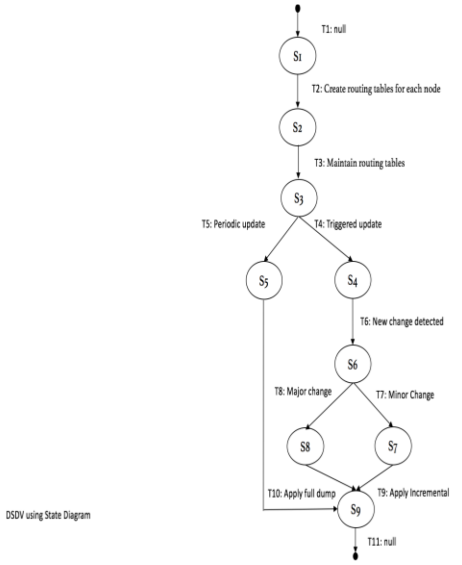


Fig. 4. DSDV using STT.

Table IV presents the states and events of DSDV, which occurred in order to move from one state to the other. Each event is described in Table V.

Table V presents a description of the DSDV events, which describes the normal functioning of the protocol. It comprises the creation of the routing protocol table and its maintenance, detecting updates and applying full dump.

TABLE IV. STATE TRANSITION TABLE FOR DSDV

	S1	S2	S3	S4	S5	S6	S7	S8	S9
S1		Event X							
S2			Event Y						
S3				Event J	Event I				
S4						Event H			
S5									Event G
S6							Event K	Event Z	
S7									Event F
S8									Event C
S9									

TABLE V. DESCRIPTION OF DSDV EVENTS

Event	**Description**
Event X	Create routing tables for each node
Event Y	Maintain routing tables
Event J	Triggered update
Event I	Periodic update
Event H	New change detected
Event K	Minor Change
Event Z	Major change
Event F	Apply Incremental
Event C	Apply full dump
Event G	Null

IV. PROPOSED APPROACH

In this section the description of the proposed algorithm, which is an improved version of the ACO algorithm is presented.

A. The proposed improvement to the ACO Algorithm

In this research work, two improvements to the Ant Colony Optimisation (ACO) algorithm are introduced in order to cover critical states with an optimal number of transitions in the state transition diagram of the protocol under test (PUT), as demonstrated in the following pseudo-code.

```
Algorithm: Ant Colony Optimization

Input: 2D State Transition Table represents the State Transition Diagram.
Output: Number of paths covered, number of visited states, number of transitions covered.

Algorithm for ant p:
Initialize all parameters
    set evaporation factor to 0.1
    set $\alpha$ to 1
    set $\beta$ to 3
    set evaporation factor to 0.1epresentumber of transitions in the s
    Set count: count= Cyclomatic complexity.
    Set key: key1=end _node, it is a variable which store the value of end node.
While (count>0)
  Evaluation at vertex 'i'
Initialize: i = start.
 Update the track: Update the visited status for the current vertex 'i'
 i.e. if (Vs[i] = =0)    then Vs[i] =1
Evaluate Feasible Set: This is to determine F (p) for the current vertex 'i'.
Evaluate the probability from the current vertex 'i' to all feasible sets s in the F (p).
The probability is calculated based on the following formula:
    $P = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum_{1}^{k}((\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta}))}$
for every k belonging to a feasible set F (p).

Move to next vertex: Using the rule below to move to the next vertex
R1: Select paths (i->j) with maximum probability (Pij).
R2: If two or more paths (from i j and i k) have equal probability like (Pij = Pik ) then select path according to below rule:
    R2.1. Compare each entry in the feasible set with the end_ node i.e.
If (j== end _node) then select 'k' as the next node otherwise follow R2.2
    R2.2. If Vs [j] =Vs [k] then select randomly
     Update track: update the covered transition for Tij, if Tij=0 then Tij=1
```

```
Start= next_node
Update the parameter Update Pheromone: Pheromone value updated for transition (i->j) according to the following rule:
If (j= = end_node)
    Go step 2
else
    τ    Go
At the end of each transition and once the ant reaches the destination vertex j the pheromone will be "evaporated" according to:
    $τij = τij - 0.1$
 If (start! = end_node)
go to step 2.3.
else count =count-1.
Go to step 2
End //End of algorithm
```

- The first is the optimization of the ant number in the start state by introducing the Cyclomatic Complexity (CC) concept. The Cyclomatic Complexity (CC) is defined as "CC= Number of closed areas in the diagram +1".

The two advantages of using Cyclomatic Complexity are that it improves the running performance time and optimizes the minimum number of ants. Simulation results show that the running time is largely reduced, compared to the traditional approach where the number of ants is computed based on the ant factory index.

- The second improvement that this research has introduced is the selection factor to critical states. The main idea is to first generate test cases for paths covering a maximum number of critical states. This way, the proposed algorithm reduces the testing effort and guarantees testing of the system's main functionalities.

- The factor P between two states *i*, and *j* is computed as follows:

$$P = \frac{(\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})}{\sum_1^k((\tau_{i,j}^{\alpha})(\eta_{i,j}^{\beta})} \qquad (1)$$

Where:

- $\tau_{i,j}$ is the pheromone value of the transition between state i and j.

- $\eta_{i,j}$ is the state priority (high = 3, normal =2, and low =1). In this case, the proposed idea will not face the scenario where there is an equal selection factor from a state which has two feasible paths, because of the existence of a pheromone value in the expression and that will vary from one transition to the other, and also because of the consideration of the evaporation factor (which is explained later). The priorities are determined based on the criticality of the transition.

- $\sum$ The summation of all possible transitions from i to all the neighbouring vertices.

- α= 1 and β=3 (give more influence on the priority states)

This means that if there are two feasible transitions with a high and low priority each, the high priority transition will be selected as shown in the selection factor formula (1). The low priority transition will be selected in the next iteration since it has not been visited. In the case of two transitions, which have been visited, the transition with the higher priority will be selected again. Each value will be discussed in more details:

a) Visited status set: Vs shows information about all the states, which are already traversed by the ant p. For any state 'i':

- Vs (i) =0 shows that vertex 'i' is not visited yet by the ant p.
- Whereas Vs (i) =1 indicates that state 'i' is already visited by the ant p.

b) Feasible set: The procedure evaluates the entire possible transition from the current vertex i to the all the neighboring vertices with the help of a state transition diagram. For example, if there is a vertex i connected to j and k, this means that there are two possible feasible transitions from vertex i§. Otherwise, the algorithm will terminate if there is no feasible path from the current vertex i.

c) Pheromone value**:** The pheromone value will be incremented by 1 in each transition using the following formula: $\tau ij = \tau ij + 1$, and by the end of each transition the pheromone will be evaporated by this rule:

When the ant moves from i to j the pheromone value will be increased at this moment using this formula: $\tau ij = \tau ij + 1$, but when the ant reaches to j the pheromone value will be evaporated using $\tau ij = \tau ij - 0.1$. The pheromone value will also be evaporated for previous events that the ant passes through. For example, if there is a transition from S1 to S2 and from S2 to S3 the evaporation is calculated as:

S1 $\longrightarrow$ S2

$\tau ij = 2$ at S2 the pheromone will be evaporated

$\tau ij = 2 - 0.1 = 1.9$

S2 $\longrightarrow$ S3

$\tau ij = 1.9 + 1 = 2.9$, the pheromone will be evaporated

$\tau\ ij = 2.9 - 0.1 = 1.8$

The pheromone will also be evaporated for S1 and S2

$\tau\ ij = 1.9 - 0.1 = 1.8$

d) The evaporation factor: The evaporation will be calculated to avoid conflict in event selection when priorities are equal.

V. Results

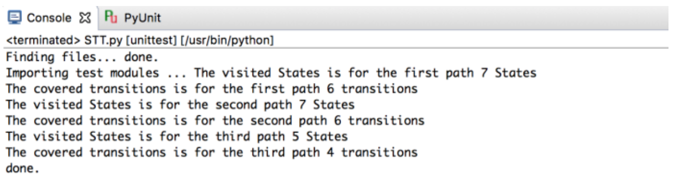
As stated in Section III-A, one of the steps in our proposed approach is protocol formalization, which is implementing the state machine formalization model to a specific routing protocol. In this research, state transition testing on the wireless network routing protocol DSDV is applied.

The state transition diagram for DSDV contains three paths (3), nine states (9), and ten transitions in total (10). Table VI shows the number of states, transitions, and paths of the protocol under test.

TABLE VI. Number of States, Transitions, and Paths in DSDV

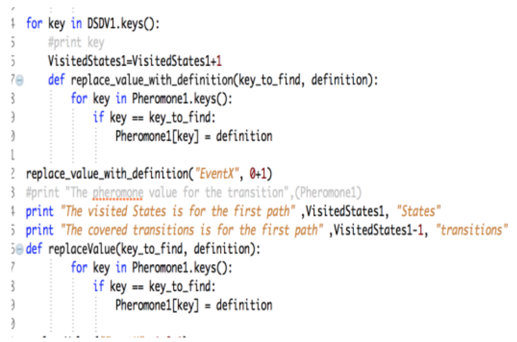
State transition components	Values
States	9 states
Transitions	10 transitions
Paths	3 paths

As stated in the test criteria, Fig. 5 shows the output after implementing the proposed algorithm in Python. The output shows the number of visited states for each path and the number of covered transitions. Fig. 6 illustrates how the automation testing is done and how the pheromone value is updated.



```
<terminated> STT.py [unittest] [/usr/bin/python]
Finding files... done.
Importing test modules ... The visited States is for the first path 7 States
The covered transitions is for the first path 6 transitions
The visited States is for the second path 7 States
The covered transitions is for the second path 6 transitions
The visited States is for the third path 5 States
The covered transitions is for the third path 4 transitions
done.
```

Fig. 5. The output after applying the proposed algorithm.



```
for key in DSDV1.keys():
    #print key
    VisitedStates1=VisitedStates1+1
    def replace_value_with_definition(key_to_find, definition):
        for key in Pheromone1.keys():
            if key == key_to_find:
                Pheromone1[key] = definition

replace_value_with_definition("EventX", 0+1)
#print "The pheromone value for the transition",(Pheromone1)
print "The visited States is for the first path" ,VisitedStates1, "States"
print "The covered transitions is for the first path" ,VisitedStates1-1, "transitions"
def replaceValue(key_to_find, definition):
        for key in Pheromone1.keys():
            if key == key_to_find:
                Pheromone1[key] = definition
```

Fig. 6. Part of automation process.

Table VIII shows the number of visited states, covered transitions, and paths.

TABLE VII. The Coverage after Applying the Proposed

Variables	Coverage
States	19
Transitions	16
Paths	3

Fig. 7 illustrates the coverage on the three levels: states, transitions, and paths as described in Table VII.

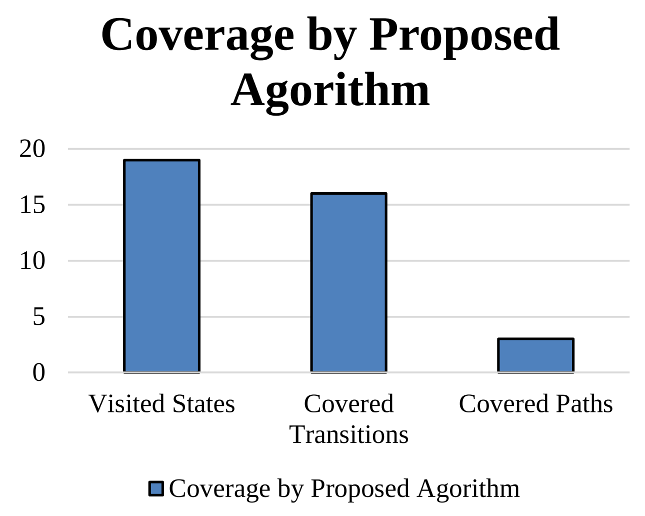


Fig. 7. The coverage of DSDV using the proposed algorithm.

This research also applies the normal ACO algorithm on the DSVV protocol in order to compare results from both approaches. Table IX shows the coverage of states, transitions, and paths.

TABLE VIII. The Coverage after Applying the Normal ACO

Variables	Coverage
States	19
Transitions	16
Paths	3

The number of visited states, covered transitions, and paths are the same as in the proposed approach.

This paper presented the protocol under test, DSDV, and the number of states, transitions, and paths that the protocol contains. It then demonstrated the results after applying both the proposed approach and the traditional testing approach in order to compare the two.

VI. Discussion and Evaluation

A. Discussion

This research paper addresses the problems in existing state-based approaches, such as infeasible paths and the length of test sequences being too long, making the testing process too complex. The proposed algorithm based on ACO finds good paths through the graph. It not only detects feasible paths but also generates minimal non-redundant test cases by providing all definition-use paths, compared to existing state based approaches. This eliminates redundant test cases and saves time consumption.

As mentioned previously, the test completion criteria have been chosen to be as follows:

- All states have been reached at least once
- All transitions have been executed at least once
- All feasible paths have been executed

The number of visited states in each path has been visited. Note that for some states the value of the visited states is more than one, meaning that the ants have visited the state more than one time. For example, {S1, S2, S3, S9} have been visited three times, and similarly {S4, S6} have been visited twice. Other states have only been visited once. Thus, all the states have been ensured to be visited at least once and that all the transitions and feasible paths have been executed. Fig. 8 shows how all the mentioned criteria have been met. The line with "bold dashes" denotes the most critical path, then the "dashes", and finally the "dots" cover the least critical path.

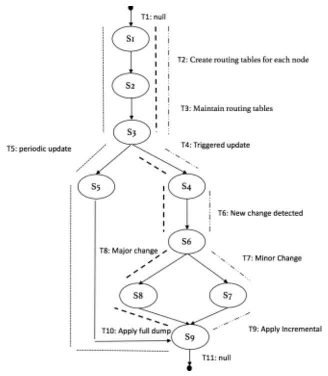


Fig. 8. The coverage of the proposed algorithm.

Furthermore, the normal ACO algorithm has been applied on the protocol in order to compare it with the output from the proposed approach. As demonstrated, the results of the proposed approach are the same as the normal ACO approach in terms of coverage due to the limited number of paths, transitions, and states that the DSDV protocol contains.

B. Evaluation

This section presents a comparison of the results between the normal ACO and the proposed approach. Table IX shows the results of both approaches.

TABLE IX. COMPARISON BETWEEN THE NORMAL ACO AND THE PROPOSED APPROACH

	The normal ACO	**The proposed approach**
States	19	19
Transitions	16	16
Paths	19	3

As illustrated in Table IX there is a remarkable difference in terms of the number of paths and coverage parameters. The system's performance was enhanced compared to the normal ACO. Thus, the proposed algorithm ensures that minimal number of paths is used which enhances the performance of the algorithm.

VII. CONCLUSION AND FUTURE WORK

The proposed approach is an improvement to the ACO algorithm. The modification was applied on the selection factor and included the priorities of each event to be calculated with the pheromone value, in order to help the ants in making their decisions. The use of α and β ensures to give the priority states more influence.

As a result, the approach reduced the number of generated test cases and provided full state and transition coverage. The test completion criteria were: all states reached at least once, all transitions executed at least once, and all feasible paths being executed.

Moreover, normal ACO testing has been applied on the protocol in order to perform a comparison between the results. The results showed that the number of visited states, covered transitions, and paths are the same as the results from the proposed algorithm.

As a future work, two ideas are to be proposed: the first is to apply the proposed algorithm on another complex protocol to see the differences in results between the automated testing and the traditional one. The second is to allow users to interact with the software by developing a graphical user interface, which adds animations to the output.

REFERENCES

[1] S. Biswas, "Applying Ant Colony Optimization in Software Testing to Generate Prioritized Optimal Path and Test Data," 2nd Int'l Conf. Electr. Eng. Inf. Commun. Technol. 2015, no. May, pp. 21–23, 2015.

[2] S. Seth and M. A. Venkatesulu, TCP/IP Architecture, Design and Implementation in Linux, PDF. Wiley-IEEE Press, 2009.

[3] J. Heidemann, K. Mills, and S. Kumar, "Expanding confidence in network simulations," IEEE Netw., vol. 15, no. 5, pp. 58–63, 2001.

[4] P. R. Srivastava and K. Baby, "Automated Software Testing Using Metahurestic Technique Based on an Ant Colony Optimization," Electron. Syst. Des. (ISED), 2010 Int. Symp., 2010.

[5] "Software Testing - Quick Guide." .

[6] A. Arya and J. Singh, "Comparative Study of AODV , DSDV and DSR Routing Protocols in Wireless Sensor Network Using NS-2 Simulator," vol. 5, no. 4, pp. 5053–5056, 2014.

[7] A. Khandakar, "Step by step procedural comparison of DSR, AODV and DSDV routing protocol," Int. Proc. Comput. Sci. …, vol. 40, no. Iccet, pp. 36–40, 2012.

[8] B. Jagdale, "Analysis and Comparison of Distance Vector,DSDV and AODV Protocol of MANET," Int. J. Distrib. Parallel Syst., vol. 3, no. 2, pp. 121–131, 2012.

[9] R. Mavinakere, "State transition testing -," pp. 1–18.

[10] T. S. Marco Dorigo, Mauro Birattari, "Ant Colony Optimization . A Computational Intelligence Technique," IEEE Comput. Intell. Mag., vol. 1, no. 4, pp. 28–39, 2006.

[11] T. Shu, M. Gu, and J. Xia, "An Evolving-Graph-Based Finite State Machine Model for Protocol Conformance Testing in MANETs," vol. 10, no. 10, pp. 359–368, 2015.

[12] O. Banias and D. I. Curiac, "Wireless Sensor Network software testing framework," Comput. Cybern. Tech. Informatics (ICCC-CONTI), 2010 Int. Jt. Conf., pp. 517–521, 2010.

[13] J. Bieniasz, P. Sapiecha, M. Smolarczyk, and K. Szczypiorski, "Towards model-based anomaly detection in network communication protocols," 2016.

[14] OPNET, "OPNET Technologies – Network Simulator | Riverbed." [Online]. Available: https://www.riverbed.com/gb/products/steelcentral/opnet.html?redirect=opnet. [Accessed: 18-Dec-2016].

[15] "Qualnet - Packet Trace | SCALABLE Networks." [Online]. Available: http://web.scalable-networks.com/qualnet-network-simulator. [Accessed: 18-Dec-2016].

[16] A. ur Rehman Khan, S. M. Bilal, and M. Othman, "A Performance Comparison of Network Simulators for Wireless Networks," 2012 IEEE Int. Conf. Control Syst. Comput. Eng., pp. 34–38, 2012.

[17] M. Balouchestani, K. Raahemifar, and S. Krishnan, "Increasing the reliability of wireless sensor network with a new testing approach based on compressed sensing theory," 2011 Eighth Int. Conf. Wirel. Opt. Commun. Networks, pp. 1–4, 2011.

[18] K. Fall and K. Varadhan, "The network simulator (ns-2)," URL http//www. isi. edu/nsnam/ns, 2007.

[19] nsnam.org, "Documentation « ns-3," 2011. [Online]. Available: https://www.nsnam.org/documentation/. [Accessed: 21-Dec-2016].

[20] J. Heidemann and U. S. C. Isi, "OMNeT++ Discrete Event Simulator," Audio, no. March, pp. 1–9, 2002.

[21] E. Weingärtner, H. Vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in IEEE International Conference on Communications, 2009.

[22] Lessmann, P. Janacik, L. Lachev, and D. Orfanus, "Comparative Study of Wireless Network Simulators," Seventh Int. Conf. Netw. (icn 2008), pp. 517–523, 2008.

[23] R. Alena, D. Evenson, and M. Rundquistt, "Analysis and Testing of Mobile Wireless Networks."

[24] S. Ji, Q. Pei, Y. Zeng, C. Yang, and S. P. Bu, "An automated black-box testing approach for WSN security protocols," in Proceedings - 2011 7th International Conference on Computational Intelligence and Security, CIS 2011, 2011, pp. 693–697.