P-ISSN : 2715-2448 | E-ISSN : 2715-7199
Vol.4 No.1 January 2023
Buana Information Technology and Computer Sciences (BIT and CS)

An Enhanced Bio-Inspired Aco Model For Fault-Tolerant Networks

Samuel W Lusweti [1]
Masinde Muliro University of Science and Technology
Email: lusweti015@gmail.com

Collins O Odoyo [2]
Masinde Muliro University of Science and Technology
Email: codoyo@mmust.ac.ke

Dorothy A Rambim [3]
Masinde Muliro University of Science and Technology
Email: drambim@mmust.ac.ke

Abstract—This research mainly aimed at establishing the current functionality of computer network systems, evaluating the causes of network faults, and developing an enhanced model based on the existing ACO model to help solve these network issues. The new model developed suggests ways of solving packet looping and traffic problems in common networks that use standard switches. The researcher used simulation as a method of carrying out this research whereby an enhanced algorithm was developed and used to monitor and control the flow of packets over the computer network. The researcher used an experimental research design that involved the development of a computer model and collecting data from the model. The traffic of packets was monitored by the Cisco Packet Tracer tool in which a network of four computers was created and used to simulate a real network system. Data collected from the simulated network was analyzed using the ping tool, observation of the movement of packets in the network and message delivery status displayed by the Cisco Packet Tracer. In the experiment, a control was used to show the behavior of the network in ideal conditions without varying any parameters. Here, all the packets sent were completely and correctly received. Secondly, when a loop was introduced in the network it was found that the network was adversely affected because for all packets sent by the computers on the network, none of them was delivered due to stagnation of packets. In the third experiment, still, with the loops on, a new ACO model was introduced in the cisco packet tracer used to simulate the network. In this experiment, all the packets sent were completely and correctly delivered just like in the control experiment.

Keywords—ACO, packets ,Loops, Networks, Algorithm

Abstrak—Penelitian ini bertujuan untuk menetapkan fungsionalitas sistem jaringan komputer saat ini, mengevaluasi penyebab kesalahan jaringan, dan mengembangkan model yang ditingkatkan berdasarkan model ACO yang ada untuk membantu memecahkan masalah jaringan ini. Model baru yang dikembangkan menunjukkan cara-cara untuk memecahkan perulangan paket dan masalah lalu lintas di jaringan umum yang menggunakan standard switches. Peneliti menggunakan simulasi sebagai metode untuk melakukan penelitian ini dimana algoritma yang disempurnakan dikembangkan dan digunakan untuk memantau dan mengontrol aliran paket melalui jaringan komputer. Peneliti menggunakan desain penelitian eksperimental yang melibatkan pengembangan model komputer dan mengumpulkan data dari model tersebut. Lalu lintas paket dipantau oleh alat Cisco Packet Tracer di mana jaringan empat komputer dibuat dan digunakan untuk mensimulasikan sistem jaringan nyata. Data yang dikumpulkan dari simulasi jaringan dianalisis menggunakan alat ping, pengamatan pergerakan paket dalam jaringan dan status pengiriman pesan yang ditampilkan oleh Cisco Packet Tracer. Dalam percobaan, kontrol digunakan untuk menunjukkan perilaku jaringan dalam kondisi ideal tanpa memvariasikan parameter apa pun. Semua paket yang dikirim diterima dengan lengkap dan benar. Ketika sebuah loop diperkenalkan pada jaringan, ditemukan bahwa jaringan terpengaruh secara negatif karena untuk semua paket yang dikirim oleh komputer di jaringan, tidak ada satupun yang dikirim karena stagnasi paket. Dalam percobaan ketiga, masih dengan loop aktif, model ACO baru diperkenalkan di pelacak paket cisco yang digunakan untuk mensimulasikan jaringan. Dalam percobaan ini, semua paket yang dikirim dikirim dengan lengkap dan benar seperti pada percobaan kontrol.

Kata Kunci—ACO, packets ,Loops, Networks, Algorithm

I. INTRODUCTION

One can rarely do anything with data that doesn’t involve a computer network since these networks enable us to share information and other resources [1]. However, these networks have to be maintained well to keep providing users with these functionalities. These computer networks can fail to work especially when a network device fails or the communication link malfunctions or is being overused against its capacity [2]. The existing networks are more complex than the conventional networks therefore it becomes hard to create, install, manage and keep them up and running efficiently all the time [3]. Due to these challenges, new technologies are needed to be employed in managing these dynamic networks that are at the center of business transactions today. There exist similar problems in real-life situations and their biological solutions which are naturally evolved and can also be applied in networking paradigms to help curb the drawbacks [3]. These are commonly referred to as Bio-inspired systems. A bio-inspired system depicts a strong relationship between a proposed algorithm aimed at solving a certain problem and a biological or natural system possessing similar capabilities [4]. There exists a necessity to employ bio-inspired systems in computer networks because living organisms like the ant colonies look better organized in their daily activities than the current internet [4]. This is because of the resilience to failure by biological systems to both internal and external environmental factors [5]. There exist many optimization algorithms including Particle Swarm Optimization, genetic algorithm, leaping frog among others. In general, ACO is the most popular and most successful algorithm that has ever been used in combinatorial optimization problems [6].

II. LITERATURE REVIEW

2.1. Loops

Loops can occur in a network whenever there exists more than one path (redundant links) at layer 2 between two endpoints or there are multiple connections between any two switches in the network or there is a physical connection between two active ports of the same switch. This loop creates broadcast storms during the forwarding of broadcasts and multicasts by switches. These switches repeatedly rebroadcast the message hence flooding the network [7].

This means that packets that are sent along a given path will eventually be stuck in that network forever making cycles, [8] unless some mechanisms are invoked which will help in flushing such packets out of the cycle. Whenever you are using link-state protocols for instance OSPF, the forwarding loops may transiently occur if the routers in use adapt their own forwarding route tables in response to a change in topology [9]. The adverse effects of the loop on Ethernet switched networks include a reduction in bandwidth, memory clogging, and packet loss [7]. These and many other challenges in computer networks need to addressed using more intelligent mechanisms especially the use of metaheuristics like bio-inspired systems. This research paper concentrates on packet loops and provides an enhanced bio-inspired mechanism to help solve the problem.

2.2 Bio-inspired systems

Biology is frequently being employed as an inspiration for research in computer science [2] and other fields like engineering, mathematics, energy [10], and business. Ant algorithms are in use today mainly to solve optimization problems instead of the problems and challenges that they were originally or initially developed to solve [11]. A direct approach to getting a solution to the combinatorial optimization problems is an exhaustive search [12], whereby the agents in these biological systems whose analogy is used to create bio-inspired systems, enumerate all possible solutions and choose the best one. The method of ants which is a bio-inspired system has proved to outshine other general-purpose algorithms for optimization like the genetic algorithms. This is evident especially when employed in combinatorial optimization problems which require the interaction of cooperating agents [11]. A good number of such intelligent algorithms are therefore being developed with the aim of solving various complex problems. Whereas some studies try to explore the application of bio-inspired algorithms theoretically, others are continuously working to improve the functionality of the algorithms [10]. This becomes a green light for the growth and development of artificial intelligence systems which mimic the behavior of ants, especially during foraging. These algorithms include Neural Networks, Particle Swarm, Genetic Algorithm, and Ant Colony Optimization algorithm among others [10].

2.2.1 ACO architecture and design

During the early years of 1990s, there was an introduction of the ACO model proposed by M. Dorigo and his companions as a metaheuristic optimization algorithm that is naturally inspired for solving hard combinatorial optimization problems [13]. ACO is in the category of metaheuristics which are probabilistic algorithms for obtaining good solutions to hard combinatorial optimization problems with a reasonable time of computation [14]. The other examples of metaheuristics include tabu-search, evolutionary computation, and simulated annealing [15] [16]. The foraging behavior of real ants inspired the creation and deployment of ACO in various fields. ACO is a very popular and modern optimization paradigm that gets its motivation from the scenario of how ant colonies find the shortest routes between their home and the food source [6]. During the process of searching for food (foraging), the forward ants start by randomly exploring the environment that surrounds their nest before they locate the source of food and deposit pheromone trails [17]. Despite its well-known popularity and application, the theory of ACO is still under development and is in its infancy stages; therefore a solid foundation of its theory is required [18]. The ACO model has been modified in many forms since its introduction. However, these forms have a common architecture. The following figures show an image of real ants in a colony foraging and a flow chart of activities carried out by ants in an ACO algorithm during their foraging activities.

2.2.2 How ACO works

Ants are self-organized biological systems exhibiting three main principles of self-organization mechanisms which include interaction between individuals, feedback loops, and local state evaluation [19]. ACO algorithm works following an indirect communication among the simple agents of a colony, known as artificial ants, enabled by their artificial pheromone trails as their media of communication [20]. The foraging ants thus communicate by laying pheromone chemicals on the ground as they search the environment for food [18]. The other ants are consequently attracted by the laid pheromone trails and therefore tend to closely follow previous ants. In the case whereby the foraging ants discover different routes between the nest and a source of food, the shortest path typically gets filled with pheromone quicker than the longer path [18]. A fascinating aspect of ants is their ability to find the shortest routes to the source of food. This is made possible only by the ability of these ants to follow the laid down pheromone trails by the predecessor ants taking in mind that most ants are almost blind meaning visual aspects are not in use [21]. The more the ants take the shortest route, the more pheromone gets deposited, until almost all ants follow the shortest path [18].

2.3 Application of ACO in Networks

There have been many areas in which bio-inspired systems have been successfully applied, some of which have been mentioned in the sections above. However, a few of them have been applied in computer networks to enhance their functionality and improve their resilience to faults. The main area of concern being adaptive routing. A good example is the ACO algorithm for network analysis and adaptive routing

[10]. Many issues in networking are formulated as multidimensional optimization problems. As dimensions of networks are increased both in terms of the number of nodes and spatially, the centralized control of communication in these networks becomes impractical. In comparison with biological systems, an individual alone is of less interest compared to the collective behaviour of the system of a larger number of the same individuals like the ant colonies [10]. As a result, Bio-inspired systems like ACO have been developed and successfully applied in network node research and design due to the appealing analogies existing between biological systems and large computer networks [2]. ACO has been applied in computer network routing problems in different formats such as AntNet, AntHocNet, ACR [6], and Ant-BasedControl(ABC) [22] among other algorithms. ABC was the first routing algorithm applied in circuit-switched networks for instance telephone wire networks [23]. This algorithm was deployed on a simulated version of the British Telecom network, which formed the basis [20] of more research on this area of network routing problems. A highly successful application of the ACO to the dynamic routing problems is the ANTNET algorithm, which was proposed by Marco D and Di Caro [24] [25] [26]. This ANTNET being successful was recommended and applied in packet-switched networks in this case the internet for adaptive routing [20]. Later on, the ANTNET became widely useful in mobile ad-hoc networks, to solve the routing problems and was used as ANTHOCNET posting exemplary results [27]. ANTNET [28] and ANTHOCNET [29] are two today very well-known ACO-based routing algorithms. ANTNET is a proactive routing algorithm while the ANTHOCNET is a reactive routing algorithm. They possess a very high delivery rate and find paths whose lengths are very close to the length of the shortest route [28],[30].

2.4 Challenges faced by ACO

If ACO is used in network routing, the packets mimic the real ants. However, it still has some challenges. In this case, a problem arises when an ant (packet) is stuck in a cycle (loop) and is forced to revisit an already visited node. These loops bring about the stagnation of ants [31] and are undesirable in networks because they cause packet latencies [6] and the packets caught in a loop are eventually destroyed when ACO is implemented. The large number of routes or paths makes it complex to manage the routing tables while concurrently increasing the probability of having packet loops [32]. Although research was done and a new ACO was developed known as MACO, the research does not claim that MACO will fully eradicate stagnation in ants, thus offering a basis for more research to be done on ACO [31]. These reasons necessitated research on how best to re-route the packets that are caught into a loop and stagnated without destroying them and by preventing stagnation and still making them reach their intended destination in the new ACO algorithm.

III. METHOD

A model was developed using python and pygame simulator to show the movement of ants. This study used the agile family as a model of SDLC since these methods are meant to quickly adapt to changing requirements, and minimize costs of production while upholding the quality of the software under development [33]. Agile is a combination of both incremental and iterative types of SDLC [34].

IV. RESULTS AND DISCUSSION

In this chapter, the researcher presents the results of the simulation experiments performed. In these experiments, the researcher used the Cisco Packet Tracer simulator to test various configurations of the network as shown below. First of all, the packet tracer is launched and a simple network configuration of four computer devices (two desktop computers and two laptops) are configured in the network. A router is used to bridge between two different network classes each having two computers. These classes include B having a default gateway of 172.16.0.1 and class C with a default gateway of 192.168.0.1. These computers are interconnected on the network using two 24-port. The so-developed network of computers is then subjected to different conditions and tested as shown by the screenshots below.

4.1. Simple-ACO

The old model called Simple-ACO was applied in various fields including computer networks. S-ACO ants were used to implement loop elimination which in turn improve the performance of the system [12]. While moving backwards through the path that in in their memory, the ants update the pheromone concentration on the paths they pass through. The following example shows how ants in S-ACO eliminate loops as using backward pass presented by Marco Dorigo [12].

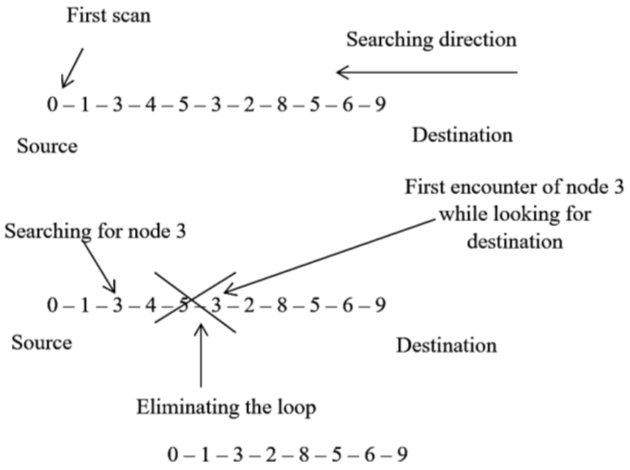


Figure 1. Final route, no loop but also node 4 is removed

In ACO, the private memory of ant is used to guarantee the probability of an ant building a feasible solution. However, during the process of finding these feasible routes, loops may appear making the ant rotate in a cycle for long. As a result, the ant ends up wasting time and resources [6]. When trying to avoid these loops, if an ant is forced to go back to an already toured node, the nodes having that cycle are removed from the private memory of the ant and information about the nodes is completely destroyed. If an ant moves in a loop for a time which is greater than half its age (Time To Live), the ant is consequently destroyed [6]. The model describes the movement of an ant from source to destination. From the flow diagram, if an ant is forced to return to an already visisted node, the node or ant will be destroyed. This saves the system from staying in a loop forever, but the killed ant never reached its destination. If the node is detryoed, then it is removed from the network making the node lose network connection.

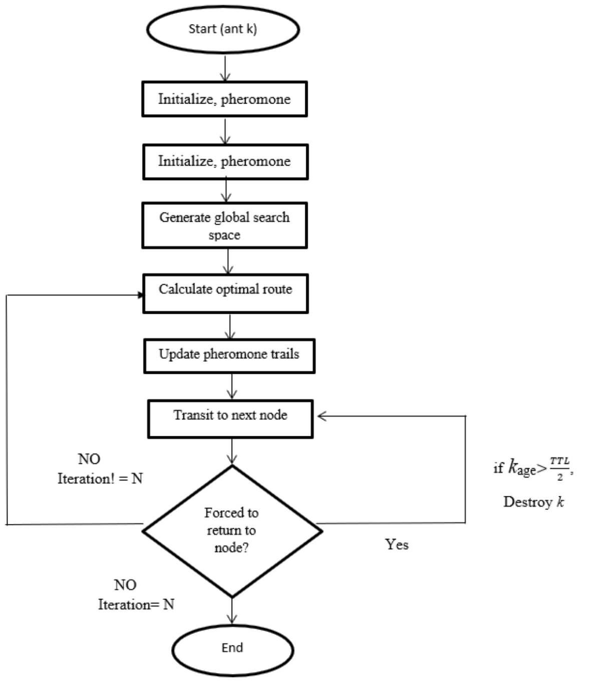


Figure 2. Flow chart of Original ACO model showing how to eliminate loops

pseudocode

```
if (k ∈ V )/∗  CHECK IF THE ANT IS IN
A LOOP AND   REMOVE IT ∗/
hops_cycle ← get cycle_ length(k,V);
     hops_fw ← hops_fw – hops_cycle;
else
     hops_fw ← hops_fw + 1;
     V [hops_fw] ← k;
     T ,[hops_fw] ← Tk→n;
end if
```

4.2. Enhanced ACO model showing physical movement of ants

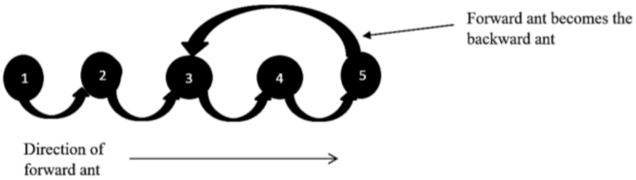


Figure 3. Loops formed are assumed to be non-existent (for instance node 4 is not removed)

In the figure above, if a loop is formed in the modified ACO model, it will be assumed by the ants as the ants will still be reroute out of the loop if they are caught in the loop for more than half of their TTL. In this case no ant will be killed and no node will be destroyed. The ants are made to be a little more intelligent in that, when an ant is caught in a cycle, its assigned time to live will be used to determine if the ant is in a loop. If the ant takes rotates in a loop for a time interval which is greater than half of its TTL, then it is rerouted out of the loop to its intended destination.

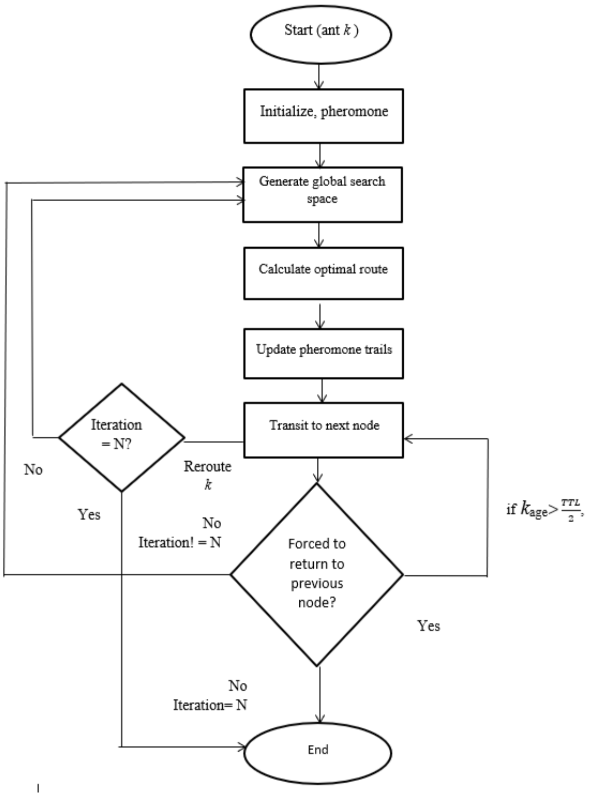


Figure 4. Flow chart Enhanced ACO model showing how to eliminate loops

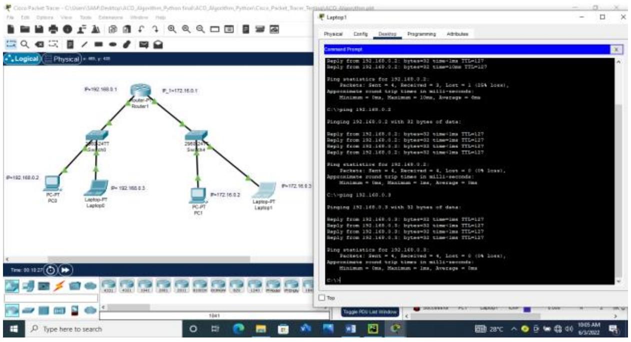
Pseudo-code of enhanced ACO model

```
if (k ∈ V )/∗  CHECK IF THE ANT IS IN
A LOOP AND REMOVE IT ∗/
 hops_cycle ← getcycle length(k,V;
      hops_fw>TTL/2;
else
hops_fw ← hops_fw + 1;
V [hops_fw] ← k;
      T ,[hops_fw] ← Tk→n;
          end if
```

In the enhanced algorithm above, if an ant is caught in a cycle, hops_fw>TTL/2 , the else statement function hops_fw ← hops_fw + 1 is invoked to get the ant out of the loop unconditionally. The ant selects the next node described in its foraging instructions. This enhanced algorithm is then applied in the simulated computer networks to help reroute packs intentionally forced to return to selected nodes. This forced return is done by introduction of forced loops.

4.0 Control experiment

Pinging the devices in the control experiment for instance from laptop 1 having IP address 172.16.0.3 to laptop 0 and PC 0 having IP addresses 192.168.0.3 and 192.168.0.2 respectively, still shows communication taking place as shown by the command prompt screen in Figure 5 below.



```
Reply from 192.168.0.2: bytes=32 time<1ms TTL=127
Reply from 192.168.0.2: bytes=32 time=10ms TTL=127

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 10ms, Average = 3ms

C:\>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data:

Reply from 192.168.0.2: bytes=32 time<1ms TTL=127
Reply from 192.168.0.2: bytes=32 time=1ms TTL=127
Reply from 192.168.0.2: bytes=32 time<1ms TTL=127
Reply from 192.168.0.2: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>ping 192.168.0.3

Pinging 192.168.0.3 with 32 bytes of data:

Reply from 192.168.0.3: bytes=32 time=1ms TTL=127
Reply from 192.168.0.3: bytes=32 time<1ms TTL=127
Reply from 192.168.0.3: bytes=32 time<1ms TTL=127
Reply from 192.168.0.3: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\>
```

Figure 5. Control results

4.1 Simulator with loops but without the algorithm

The simulated network is configured as shown below. There is a modification from the control experiment by the introduction of physical loops on the network. Figure 6 below shows the configuration.

After simulating packets in this network, the message remained in progress forever as shown in the bottom right corner of the figure below. Secondly, the messages in the network simulator kept on rotating between the loops created and the two switches 0 and 4.

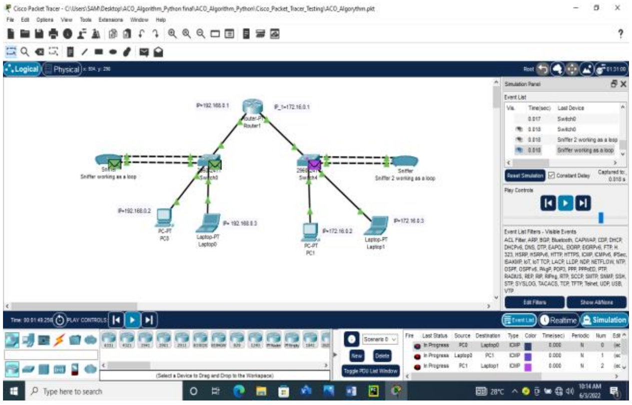
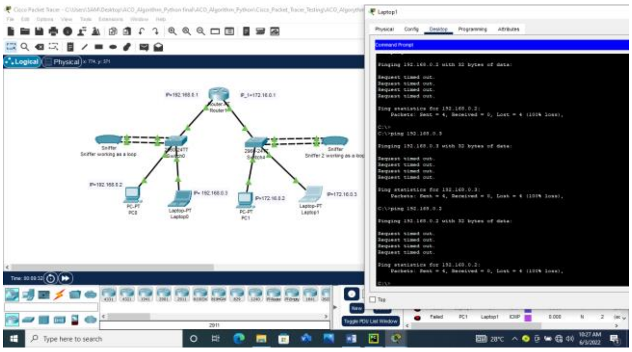


Figure 6. Loops without ACO result 1

When we open the command prompt and ping from laptop 1 having IP address 172.16.0.3 to laptop 0 and PC 0 having IP addresses 192.168.0.3 and 192.168.0.2 respectively, the ICMP message fails to be delivered and shows time out on the command prompt and failed on packet tracer as shown in figure 7 below.



```
Pinging 192.168.0.2 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>
C:\>ping 192.168.0.3

Pinging 192.168.0.3 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.0.3:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>
```

Figure 7. Loops without ACO result 2

4.2 Simulator with loops and with the algorithm

The researcher was able to run the algorithm and then simulated the network having the loops. Secondly, the main algorithm is imported in the programming mode under the TCP Python package of each computer device running in the networking. The algorithm displays the activity of random movement of ants on the pygame.

The simulation mode of the Cisco Packet Tracer shows message delivery in the network with green ticks as shown in the figure below. At the bottom right corner of the simulator, there is an indication of the successful delivery of messages.

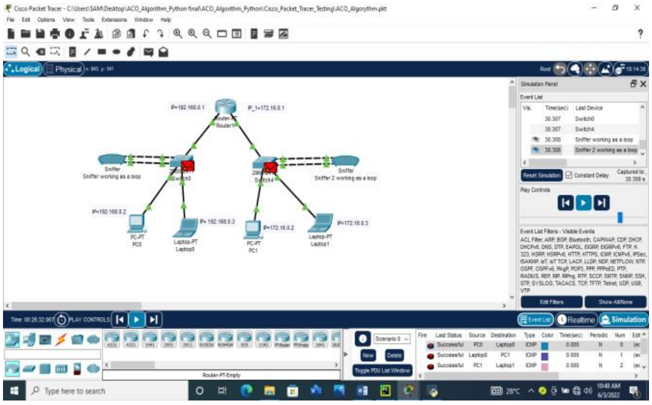
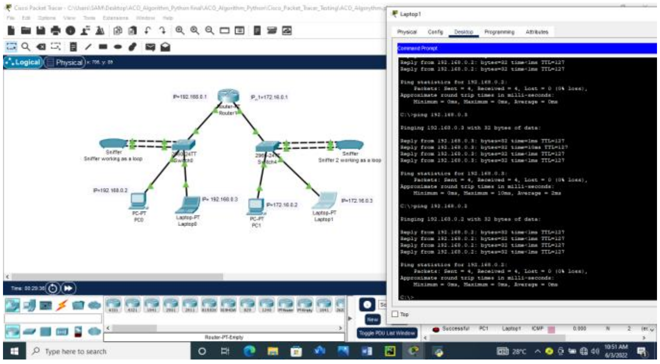


Figure 8. Loops with ACO result 1

Pinging the devices for instance from laptop 1 having IP address 172.16.0.3 to laptop 0 and PC 0 having IP addresses 192.168.0.3 and 192.168.0.2 respectively, indicates communication taking place as shown by the command prompt screen below.



```
Reply from 192.168.0.2: bytes=32 time<1ms TTL=127
Reply from 192.168.0.2: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>ping 192.168.0.3

Pinging 192.168.0.3 with 32 bytes of data:

Reply from 192.168.0.3: bytes=32 time=1ms TTL=127
Reply from 192.168.0.3: bytes=32 time=10ms TTL=127
Reply from 192.168.0.3: bytes=32 time<1ms TTL=127
Reply from 192.168.0.3: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.0.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 10ms, Average = 2ms

C:\>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data:

Reply from 192.168.0.2: bytes=32 time<1ms TTL=127
Reply from 192.168.0.2: bytes=32 time<1ms TTL=127
Reply from 192.168.0.2: bytes=32 time<1ms TTL=127
Reply from 192.168.0.2: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\>
```

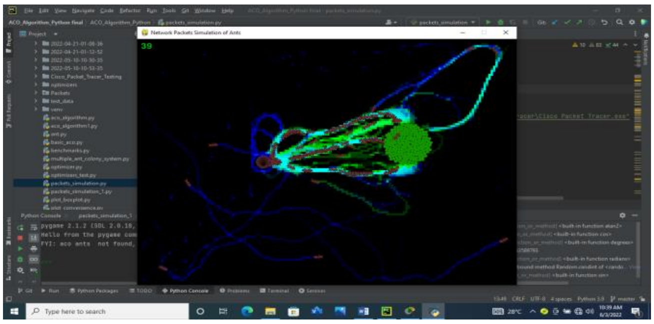
Figure 9. Loops with ACO result 2

4.4 Interpretation of results

In the above simulation, the researcher used three experiments to test and validate the data collected. The first which is a control experiment provides an ideal network with no variations in the network working under optimal conditions. It was discovered that the network messages were delivered properly between the devices on the network without any problem showing 100% success for the 4 messages sent over the network. In the second experiment, the researcher subjected the simulated network to forced loops as it normally happens in colleges and universities (mostly done by students). In this experiment, it was discovered that the network went completely down and could not recover from the looping packets which completely failed to be delivered to the intended destination. Pinging the network showed that the 4 messages send had a 100% failure rate. In the third experiment, the ACO Algorithm was simulated alongside the network with the loops still on. This experiment showed packets being delivered normally in the simulation mode by showing success to all ICMP packets sent. The ping tool also showed a 100% success rate for all the 4 test packets sent by various devices on the network. This shows that the algorithm intervened to reroute the packets that would have otherwise gotten locked up in the loop as shown in experiment 2 where the packets kept rotating in the loop thereby breaking communication between the computers on the network. The algorithm helped the network in rerouting any packets that would be caught in the loop to reach their destination. From the above experiments, five average Round Trip Times(RTT) were collected from control experiment and five from enhanced experimen. The control experiment had the following RTT as shown in figure 8 and figure 9 (6,0,0,2,0) averaging to 8/5 =1.6ms. The experiment of enhanced algorithm showed the following results as RTT as witnessed from figure 16 and figure 17 (0,0,2,0,16) avearaging to 18/5=3.6ms. Although the algorithm's results show that the packets get to destination on average 2 ms later than in the case of the control, it is still witin the allowed RTT. The 2ms delay may have been brought about by the cycling packets before getting out of the loop. For optimal network performance , studies have shown that RTT<500ms [75].

4.5 How the enhanced ACO algorithm works

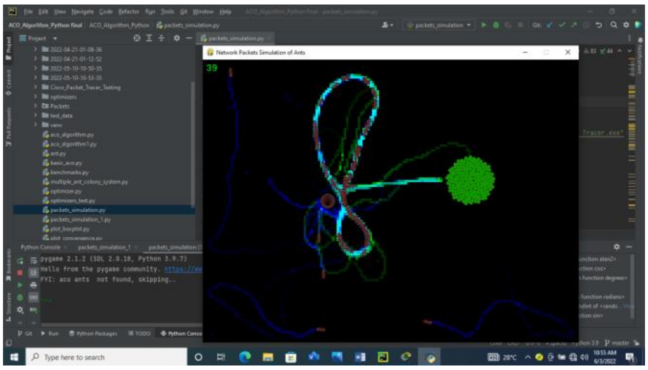
The ACO model was first developed and applied in computer networks to help the network packets reach their destination very fast using the shortest route possible by mimicking the ants through their foraging behavior and intelligence. However, the model could not solve the problem of stagnation of ants and thus the new model developed in this research helps to get the stagnated ants out of the loop. The algorithm is then imported into the Cisco Packet Tracer and the packet tracer translates the ants into real packets of the network. When the packets are caught in a loop, they use their random intelligent movement to get out of the loop, once they realize there are talking too long to reach their destination. The movement of the packets (similar to the ants after importing into packet tracer) is as shown below.



```
pygame 2.1.2 (SDL 2.0.16,
Hello from the pygame com
FYI: aco ants  not found_
```

Figure 10. Ants in model foraging

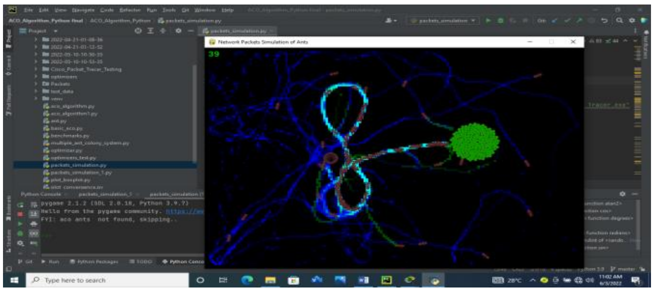
Figure 10 above shows packets finding the routes from the source of food to the nest (dest).



```
pygame 2.1.2 (SDL 2.0.16, Python 3.9.7)
Hello from the pygame community. https://ww
FYI: aco ants  not found, skipping..
```

Figure 11. Ants in model caught in a loop

The figure above shows packets (ants) caught in a loop for sometime.



```
pygame 2.1.2 (SDL 2.0.16, Python 3.9.7)
Hello from the pygame community. https://ww
FYI: aco ants  not found, skipping..
```

Figure 12. Ants getting out of the loop

This figure shows (packets) ants getting out of the loop after rotating in the loop for a few moments. The will be unnoticeable in the real network since it happens at a very high speed.

4.6 Discussion

According to research done by Kwang Hong Sim and Weng Hong Sun [31], when implementing ACO the following issues arise: stagnation of ants and poor adaptiveness of ants. Stagnation will occur if a network reaches its convergence too early (prematurely). This leads to the following problems: 1) congestion of packets (ants) on the optimal path, 2) reduction of probability of ants choosing other routes. The research further indicated that congestion and low probability of choosing other paths by ants will consequently lead to the following problems in dynamic computer networks: a) The selected path may become non-optimal if congested b) The congested path may be disconnected due to network failure. Due to these issues, Sim and Sun noted that the network would eventually have a low degree of sensitivity due to changes in topology or link failure. In their research, Sim and Sun [31] proposed a new improved model called MACO which they noted that it enhanced the adaptiveness of the network and reduced the chances of stagnation of ants. In MACO, the ants are made to deposit pheromones of different colors and observed that ants will follow routes with similar colors reducing congestion. However, they conclude that their research does not guarantee that MACO would fully eradicate stagnation of ants as it only reduced the chances of stagnation. Secondly, another research was done by Gianni Di Caro on the application of ACO to adaptive routing in telecommunications networks [8]. The research showed that loops that lead to stagnation of ants should be avoided as much as possible because they cause very high packet latencies. To implement loop avoidance, Di Caro explained using an equation for avoiding loops in networks. For instance, if an ant is forced to return to a node that it had previously visited, the nodes that composed the loop are taken out of the memory of the ant, and all information describing those nodes is destroyed. Consequently, if an ant moves on a cycle for a time that is greater than its half-life, the ant is destroyed. This destruction solves the problem of loops; although, another problem arises whereby the affected nodes and ants(packets) will be destroyed which means they will not communicate over the network as intended. Considering the finding of the above two kinds of research, the researcher carried out this study to help strike a balance between stagnation and the destruction of ants. From the results above, the problem of stagnation of ants is more highly reduced than that of MACO suggested by Sim and Sun [31] by making sure the ants that are caught in a loop simply get out of the loop using a new optimal route as shown in figure 10. This will help us make sure that all the packets in the network reach their intended destination as opposed to destroying stagnated ants as found out in the research by Di Caro [8].

V. CONCLUSION AND RECOMMENDATIONS

The results show that this research can add some contribution to the body of knowledge. When well adopted in schools, colleges, homes, and offices, the algorithm produced in this research study can go a long way in improving the reliability of the computer networks that are constantly under attack by forced loops. However, the algorithm does not completely guarantee that it will solve all the problems faced by computer networks. Further research needs to be done to improve it further.

REFERENCES

[1] Meyers, M., "Introducing Basic Network Concepts." 2010, https://www3.nd.edu/~cpoellab/teaching/cse40814_fall14/networks.pdf (Accessed June. 3, 2022).

[2] Hylsberg R. Jacobsen, Q. Zhang,T. Skjødeberg Toftegaard, "Bioinspired Principles For Large-Scale Networked Sensor Systems: An Overview", Sensors, Vol. 11, No. 4, Pp. 4137-4151, 2011. DOI: 10.3390/S110404137.

[3] Jisha Mrriam Jose, "Bio-Inspired Networking," 2011 HYPERLINK "Http://Dspace.Cusat.Ac.In/Jspui/Bitstream/123456789/3195/1/Bio-Inspired%20networking.Pdf" Http://Dspace.Cusat.Ac.In/Jspui/Bitstream/123456789/3195/1/Bio-Inspired%20networking.Pdf (Accessed Jan. 23, 2020)

[4] Jamalipour A. "Bio-Inspired Networking", The University Of Sydney: IEEE, 2009. HYPERLINK "Http://Www.It.Is.Tohoku.Ac.Jp/~Kato/Workshop2009/01.Pdf" Http://Www.It.Is.Tohoku.Ac.Jp/~Kato/Workshop2009/01.Pdf (Accessed Feb. 20, 2020)

[5] Dressler, I.F., "Self-Organization in Autonomous Sensor/Actuator Networks [SelfOrg." 2010.

[6] Caro D.G., Dorigo M. "Ant Colony Optimization And Its Application To Adaptive Routing In Telecommunication Networks, 2004". (Doctoral Dissertation, Phd Thesis, Faculté Des Sciences Appliquées, Université Libre De Bruxelles, Brussels, Belgium).

[7] Iufoaroh S.U, Chukwumaobi .O,Oranugo C.O "Tracking The Effects Of Loops In A Switched Network Using Rapid Spanning Tree Network," International Journal Of Research In Electronics And Communication Technology (IJRECT 2015), Vol. 2, No. 3, 2015.

[8] Balakrishnan H, "Network Routing - II Failures, Recovery, And Change," 2009. HYPERLINK "Http://Web.Mit.Edu/6.02/Www/S2009/Handouts/L21.Pdf(Accessed" Http://Web.Mit.Edu/6.02/Www/S2009/Handouts/L21.Pdf(Accessed Jan. 6, 2021)

[9] Francois P., Bonaventure O. "Avoiding Transient Loops During The Convergence Of Link-State Routing Protocols." IEEE/ACM Transactions On Networking, 15(6), 2007, Pp. 1280-1292

[10] Kumar K.A, "Bio Inspired Computing – A Review Of Algorithms And Scope Of Applications", Expert Systems With Applications, Vol. 59, Pp. 20-32, 2016. DOI: 10.1016/J.Eswa.2016.04.018.

[11] Afshar M.H, "A New Transition Rule For Ant Colony Optimization Algorithms: Application To Pipe Network Optimization Problems", Engineering Optimization, Vol. 37, No. 5, Pp. 525-540, 2005. DOI: 10.1080/03052150500100312.

[12] Marco D., Thomas S., "Ant Colony Optimization", London: Bradford Book, 2004.

[13] Dorigo, M., Vittorio M.,Alberto C., "Ant System: Optimization By A Colony Of Cooperating Agents." IEEE Transactions On Systems, Man, And Cybernetics, Part B (Cybernetics) 26, No. 1 (1996): 29-41

[14] Glover F., Gary A., "Handbook Of Metaheuristics." Vol. 57. Springer Science & Business Media, 2006

[15] Glover F, "Tabu Search—Part II,," ORSAJ. Comput., Vol. 2, No. 1, Pp. 4-32, 1990.

[16] Glover, F., Laguna, M.,"Tabu Search Kluwer Academic". Boston, Texas, Dordrecht, 1997.

[17] Dorigo M., Blum C., "Ant Colony Optimization Theory: A Survey," Theoretical Computer Science, Vol. 344, Pp. 243-278, 2005.

[18] Sudholt, D. And Thyssen, C., "Running Time Analysis Of Ant Colony Optimization For Shortest Path Problems". Journal Of Discrete Algorithms, 10, 2012, Pp.165-180

[19] Dressler, F., "Benefits Of Bio-Inspired Technologies For Networked Embedded Systems: An Overview". In Dagstuhl Seminar Proceedings. Schloss Dagstuhl-Leibniz-Zentrum Fr Informatik. 2006.

[20] Dorigo, M., Stützle, T.,. "Ant Colony Optimization: Overview And Recent Advances. Handbook Of Metaheuristics", 2019, Pp.311-351.

[21] Cordon O., Herrera F., Stutzle T. "A Review Of Ant Colony Optimization Metaheuristic: Basis, Models And New Trends," 2002.

[22] Mandeep Kaur, A.S "A Review On Ant Colony Optimization In MANET," International Journal For Science And Emerging, Vol. 19, No. 1, Pp. 1-6, 2014.

[23] Schoonderwoerd, R., Holland, O.E., Bruten, J., Rothkrantz,. "Ant-Based Load Balancing In Telecommunications Networks". Adaptive Behavior, 5(2), 1997, Pp.169-207.

[24] Marco. D., Gianni C., "Ant Colonies For Adaptive Routing In Packet-Switched Communications," In Fifth International Conference On Parallel Problem Solving From Nature, 1998.

[25] Marco. D., Gianni C., "Mobile Agents For Adaptive Routing.," In Proceedings Of The 31st International Conference On System Sciences, 1998.

[26] Marco. D., Gianni C., " Antnet: Distributed Stigmergic Control For Communication Networks," Journal Of Artificial Intelligence Research, Vol. 9, Pp. 317-365, 1998

[27] Gianni D., Frederick D. ,Luca M.G , "Using Ant Agents To Combine Reactive And Proactive Strategies For Routing In Mobile Ad Hoc Networks," International Journal Of Computational Intelligence And Applications, Vol. 5, No. 2, P. 169–184, 2005. DOI/10.1142/S1469026805001556.

[28] Marco. D., Gianni C., "Ant colonies for adaptive routing in packet-switched communications networks.," In Proceedings of the 5th ACM International Conference on Parallel Problem Solving from Nature, p. 673–682, 1998.

[29] Di Caro, G., Ducatelle, F., Gambardella, L.M. "AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks". European transactions on telecommunications, 16(5), 2005, pp.443-455.

[30] Gunes M., "Ara - the ant-colony based routing algorithm for manets.," In Proceedings of the 2002 International Conference on Parallel Processing Workshops, pages, pp. 79-89, 2002.

[31] Sim, K.M., Sun, W.H., November. "Multiple ant-colony optimization for network routing". In First International Symposium on Cyber Worlds, 2002. Proceedings, 2002, (pp. 277-281). IEEE.

[32] ManiezzoV., "Ant Colony Optimization", 2001.

[33] Young D., "Sofware Development Methodologies," 2013.

[34] Sharma, S., Sarkar, D., Gupta, D.,." Agile processes and methodologies: A conceptual study". International journal on computer science and Engineering, 4(5), 2012, p.892.

[35] Rasmussen J., "What is the basis for classifying a call as poor in Lync 2013 QoE?," 20 September 2013. [Online]. Available: https://docs.microsoft.com/en-us/archive/blogs/jenstr/what-is-the-basis-for-classifying-a-call-as-poor-in-lync-2013-qoe. (Accessed July. 2, 2022).