Modeling Ad-hoc Networks Using Reasoning Agents

Elena Zamyatina
HSE University
Perm, Russia
e-mail: e_zamyatina@mail.ru

Yaroslav Kirillov
Perm State University
Perm, Russia
e-mail: kirillov6@mail.ru

***Abstract*—The paper presents a software tool for agent-based simulation used to study ad-hoc networks and the results of a simulation experiment, namely, the results of executing the SDR routing algorithm. In addition, the issues of the implementation of a reasoning (intelligent) agent are considered.**

***Keywords*—simulation, agents, ad-hoc networks, routing algorithms.**

I. Introduction

Currently, there is a tendency to increase the number of mobile telecommunication devices, such as laptops, tablets, smartphones, etc. Wireless technologies are coming to the fore for building computer networks based on the principles of their self-organization. Such networks do not require any additional infrastructure other than the computational nodes themselves. In this case, all nodes take over the network management functions. There is a need to research and develop reliable and efficient protocols and algorithms, including routing algorithms. Modeling, including simulation, became the basis for the study of wireless and, in particular, Ad-hoc networks, since the modeling environment allows one to fully investigate such complex dynamic objects as mobile networks, to study new protocols, algorithms for the interaction of nodes. The use of Ad-hoc networks has several advantages over traditional wireless networks due to: (a) the ability to transmit data over long distances without increasing the transmitter power; (b) resilience of the network to changes in topology; (c) high speed of deployment. Thus, the problem posed is really urgent, an extremely important task is to be able to quickly and conveniently simulate the behavior of various algorithms, including routing algorithms in Ad-hoc networks.

There are a lot of simulation tools for the design and analysis of computer networks (OMNET ++ [1], NS-2 [2], etc.). Most of them are aimed at solving specific problems: some are more suitable for designing a network, others for analyzing a finished network. But for Ad-hoc networks, in addition to design and analysis, you need to have tools that allow you to work with graphs, with operations on them. Indeed, the dynamic graph is the mathematical model of Ad-hoc networks. The fact is that in such networks, computational nodes change their location in space, the distance between them and their relative position change. This increases the efficiency of the network: it becomes possible to have greater coverage for the communication network [3], adaptation of the network in conditions of uneven terrain, in the presence of any obstacles to the movement of nodes [4,5]. The work presents a system of computer-aided design and modeling of computer networks TriadNS [6]. Unlike other simulation tools, TriadNS has a set of standard routing algorithms and allows you to add new algorithms to existing ones. In addition, the TriadNS modeling system has linguistic tools that allow one to build and explore graphs, supports the agent-based modeling paradigm and offers linguistic tools for the implementation of a reasoning (intelligent) agent.

Further, the work is structured as follows: the features of Ad-hoc networks are briefly considered, then the DSR algorithm is described in detail, the linguistic tools of the Triad language and the results of the work of reasoning agents are presented.

II. Ad-hoc Networks and Their Features

So, the classical Ad-hoc network is a dynamic mobile network without infrastructure, which is formed automatically as a result of connections between a set of mobile nodes that are located in a certain area; in this case, centralized management is not required. In such a network, nodes are the main component; each node can function as a router, defining and maintaining routes to other nodes on the same network, as well as to an end device (transmitter or receiver). This is in contrast to wired networks and managed wireless networks, in which routers (on wired networks) or access points (on managed wireless networks) perform the task of controlling data flow. The main characteristics of Ad-hoc networks are operations with limited bandwidth and power consumption, dynamic topology, and variable bandwidth links. Minimal configuration and fast deployment allows the use of Ad-hoc networks in emergency situations such as natural disasters and military conflicts.

The routing algorithms in such networks face a number of problems, including adapting to dynamic changes and limited resources, and differ from the routing requirements in infrastructure (wired LAN and WAN) networks, although often the routing algorithms themselves remain the same, the routing protocols, which are used in infrastructure networks, turn out to be ineffective, and sometimes inoperable in the conditions of wireless Ad-hoc networks. There are already many routing protocols used in Ad-hoc networks. One of the possible options for the classification of routing protocols is based on the principles of their operation: (a) proactive; (b) reactive; (c) hybrid one.

A. Proactive protocols

The most widespread in self-organizing networks are proactive (tabular) routing protocols. Such protocols periodically send service messages over the network with information about all changes in its topology. As a result, each node in the network, based on this information, builds routes to all other nodes and stores them in the routing table, from where they are read when there is a need to send a message to any destination. Most proactive protocols use Bellman-Ford algorithms with some improvements, as well as Dijkstra's algorithm for finding the shortest route.

The use of proactive routing is most effective in sedentary and small self-organizing networks. With an increase in mobility (dynamic topology) and the number of network nodes, the use of proactive protocols leads to a rapid increase in network load with service traffic and inefficient use of energy resources of each node, which is a significant disadvantage when organizing large, dynamic networks, such as mobile Ad-hoc networks.

B. Reactive protocols

Reactive protocols create routes to specific nodes only when it becomes necessary to transfer information from the sending node to the receiving node. In such protocols, the sending node broadcasts a request message for a route, which must reach the receiving node. In response to such a message, the receiving node sends an acknowledgment message, from which the sender learns the necessary route and caches it. When re-sending data to the receiving node, the route is simply read from the cache. If the discovered route becomes unavailable, the procedure for discovering and maintaining the route is started. Reactive protocols are more efficient in dynamically changing networks due to the reduction in the amount of overhead transmitted over the network, since the search for a route is carried out only when necessary. However, there are a number of disadvantages: (a) increased delay in finding the primary route associated with high mobility and a large number of nodes; (b) finding a new path in real time, which significantly limits reactive protocols for video and voice transmission.

C. Hybrid protocols

Hybrid protocols have been proposed that combine the mechanisms of proactive and reactive protocols in large, mobile networks. Such protocols divide the network into many subnets, within which a proactive protocol operates, and the interaction between such subnets is based on reactive routing protocols. This allows you to: (a) reduce the size of the routing tables (store information about the subnet); to reduce the amount of transmitted service information. (throughout the network, since its main part is distributed only within the subnet. Consider the reactive DSR algorithm, an example of its modeling in the simulation system TriadNS.

III. Routing Algorithm DSR

Dynamic Source Routing (DSR) is a routing protocol for mobile networks, an example of a MANET with a mesh topology [5]. Explicit routing requires that the address of each node be remembered between the source node and the destination node during its lookup. So, let's call the chain of addresses of intermediate nodes a route. The information about the route is replenished with the addresses of the nodes that process the broadcast requests of the source node. This route is used to transmit packets. As a result, the routed packets contain the address of each device they passed through. Due to the explicit assignment of routes, all information about them is continuously updated by mobile nodes (as long as the data flow passes through them). This avoids the need to periodically check the route. In any case, the route is generated only if the request message has reached the source node (the chain of nodes accumulated in the request is added to the response). The DSR algorithm was designed to reduce the traffic consumed by control packets on wireless networks by eliminating table update messages. Updates to tables are needed in algorithms that form routes using tables. DSR assumes (like other reactive routing protocols) that during route formation, a node establishes a route by broadcasting a RouteRequest (RREQ) packet over the network. The receiving node, when receiving a RouteRequest packet, generates a response by sending a RouteReply (RREP) packet back to the source node. The package stores the traversed route.

A. RREQ request packet

Request initialization is performed when the source node needs to communicate with the destination node, but the nodes are not directly related to each other. The source node sends a Route Request (RREQ) to its neighbors. The request has the following structure: (a) broadcast_id - request identifier; (b) source_id - source node identifier; (c) destination_id - identifier of the destination node; (d) path - the path from the source node to the destination node; (f) hop_count - number of "hops". Each of the neighboring nodes receives a route request (RREQ) and operates in one of 2 scenarios: (1) if the node is a receiving node, then it returns a route response packet (RREP - Route Reply); (2) if a node is not a receiving node, then it forwards the RREQ packet to its neighbors.

B. RREP reply packet

If a node receives an RREQ packet and it already has a formed route to the required destination node, then it sends a response packet with this route (RREP) to the neighbor node that sent the RREQ. The structure of the RREP packet is as follows: (a) source_id - source node identifier; (b) destination_id - identifier of the destination node; (c) path - path; (d) hop_count - number of "hops". Intermediate nodes send the first RREP to the source using cached entries from the path field. Cached reverse paths will be destroyed on nodes that did not receive the RREP packet.

C. RERR error packet

When a source node has a route to a destination node, it sends a message to that node. Upon detecting a failure of one of the nodes in the route, the previous (closer to the source) node sends an error message in the route (RERR - Route Error). The structure of the RERR package is the same as that of the RREP package. If an error is detected, the hop_count field is filled with a negative number. Let's give an example:

Node 1 needs to send a data packet to node 7. Suppose that node 6 knows the current route to node 7. We also assume that there is no other information regarding the route to node 7 in the network (Fig. 1.).

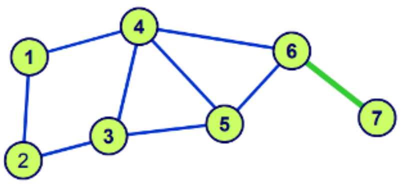
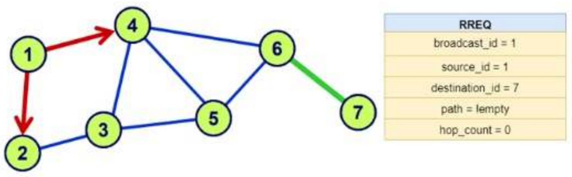


Fig.1. Node 1 needs to send a data packet to node 7

Node 1 sends an RREQ packet to neighboring nodes (Fig.2.).



RREQ
broadcast_id = 1
source_id = 1
destination_id = 7
path = lempty
hop_count = 0

Fig.2. Demonstration of the structure of an RREQ packet and the process of sending it.

Nodes 2 and 4 confirm that this is a new RREQ. But since none of the nodes is a destination node, then these nodes send forward the RREQ to all neighbors, having previously increased the hop_count by 1 and adding their ID to the path field (Fig. 3.).

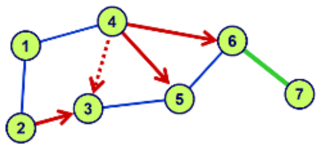
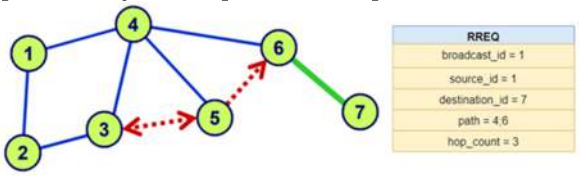


Fig.3. Nodes send RREQ to neighbors

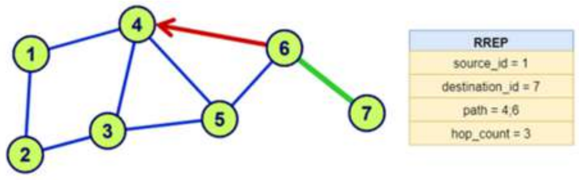
The RREQ reaches node 6, which knows the route to node 7 (Fig. 4). Nodes 3 and 5 will forward the RREQ packet, but the recipients recognize the packet as a duplicate.



RREQ
broadcast_id = 1
source_id = 1
destination_id = 7
path = 4,6
hop_count = 3

Fig.4. Nodes 3 and 5 will forward the RREQ packet

Because node 6 knows the path to node 7, then it sends an RREP packet to node 4 (see Fig. 5.).



RREP
source_id = 1
destination_id = 7
path = 4,6
hop_count = 3

Fig.5. Node 7 sends an RREP packet to node

Node 4 verifies that this is a new route response and sends an RREP packet to node 1 (Fig. 6).

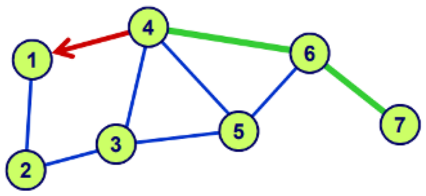


Fig.6. Node 4 verifies a new route response and sends an RREP packet to node 1

Now node 1 knows the path to node 7, consisting of 3 "hops", and can immediately use it to forward data packets (Fig. 7.)

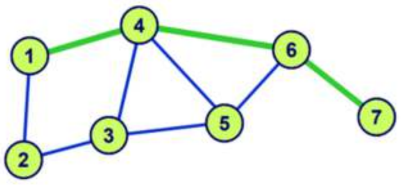


Fig.7. Now node 1 knows the path to node 7

IV. Intelligent Agents

For modeling self-organizing computer networks, the agent-based paradigm of simulation modeling is best suited, which most adequately reflects the object under study. It is known that an agent is a hardware or software entity that functions in the external environment, interacting with it and with other agents.

As already mentioned above, the TriadNS simulation tools, developed by employees of the Perm and Krasnodar Universities, have linguistic tools (Triad language) and software that are designed to describe computer networks. A computer network is represented by a graph, each vertex of which is a computational node. To describe the behavior of a computing node, a linguistic construction is used, which is called routine. Essentially, the simulation model in TriadNS is a network of interacting agents.

Currently, there is a tendency when devices operating in the network become "intelligent": they can adapt to changing environmental conditions, change their behavior, and make decisions. Such software agents are usually called intelligent. It is logical that the linguistic and software tools of simulation systems should make it possible to describe the behavior of intelligent agents using one or another knowledge model and implementing inference mechanisms. In this paper, the authors propose to use the rule-based model of knowledges. Below are the syntax constructions of the Triad language for an intelligent agent.

The syntax for a new type of routine is:

IRoutine <Name> (<Pole declaration>) [<Parameter section>] {<Initialization section>} {<Description of event and rules>} ***EndIRout***

The syntax for describing the behavior of an intelligent agent practically does not differ from the syntax of the usual routine in the Triad language, with the exception of the keywords ***IRoutine*** and ***EndIRout***, which are responsible for the beginning and end of this construction. The *initialization* section describes the facts of the knowledge base. The fact has the following format:

<Fact> :: = <Type> <Name> {: = <Value>};

A fact can have a predetermined meaning, or it can be unknown and its meaning can be obtained as a result of the operation of the inference mechanism. An example of a fact description: ***Integer*** CurMonth: = 6.

Next, we will consider how one can describe the events and rules by which the agent acts. In the section {<Description of event and rules>}, one may receive and send messages (syntax construction <event>). As in the production model of knowledge, the inference engine considers the fulfillment of each rule and performs certain actions: changing the value of a fact, scheduling an event, etc.

To describe the rules, the syntax construction <Rule> has been added to the Triad language:

<Rule> :: = ***Rule*** <Name> ***If*** <Condition> ***Then*** <Action> ***EndIf*** {Reason <Cause>} ***EndRule***.

In the description of the rule, you can indicate the reason (a verbal description of why the rule worked), then, if the agent executes the action of this rule, the reason for the execution of the rule will be indicated. Let's consider an example:

```
Rule IsSummer
        If (CurMonth = 6) | (CurMonth = 7) | (CurMonth
= 8) Then
                Schedule StartVacation; (* We schedule
an event: a beginning of the vacations*)
        EndIf
        Reason "Now is a June, a July or an August"
EndRule
```

Ontologies are used to store the knowledge base in TriadNS. To define new network elements (for Ad-hoc networks) it is necessary to add new subclasses to the existing ontology, namely, to ComputerNetworkNode class. The behavior of computer network modeling objects is determined by routines that are stored in the ComputerNetworkRoutine class. To set the behavior for the nodes of the network (routine), it is necessary to add a new instances of the routine for each node represented at Fig.8.

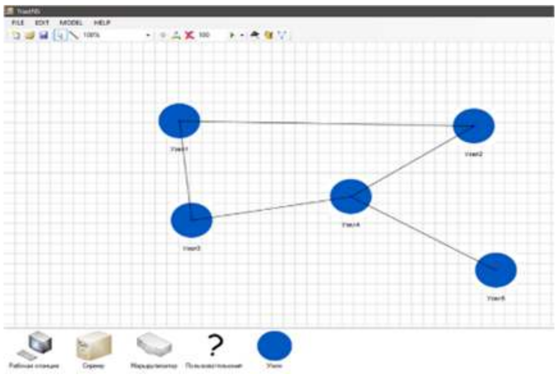


Fig. 8. Computer network structure

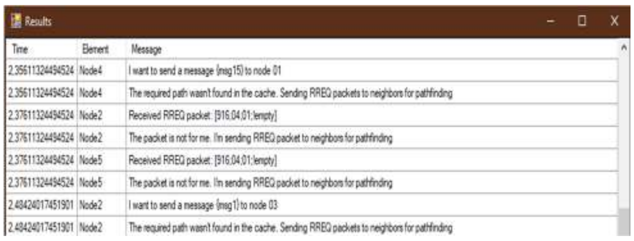
Intelligent agents, unlike reactive ones, have memory and can perform complex reasoning. Since the DSR routing protocol uses a route cache, the agent's memory is a key parameter when choosing a route. Therefore, it is intelligent agents that are suitable for the implementation of this protocol.

The DSR protocol routine sends messages to a randomly selected host on the network. When a node receives a packet, its processing and further operation of the node depends on the type of packet - RREQ, RREP or RERR. The routine has several parameters: (a) NodeID - node identifier; (b) DestinationID - the identifier of the destination node, if it is necessary to send messages to a specific node; (c) MaxCacheLen - the maximum capacity of the route cache; (d) ReqPeriod - time interval after which nodes send messages.

An example of describing the rules for implementing a routine for DSR:

```
IRoutine
…
rule RREQ
  if (tmpRecMsg[0] = "RREQ") then
    schedule WorkRREQ in 0.01;
  endif;
  reason "Received RREQ packet";
endrule;
rule RREP
  if (tmpRecMsg[0] = "RREP") then
    schedule WorkRREP in 0.01;
 endif;
reason "Received RREP packet";
endrule;
…
Endrout
```

A fragment of the log with the results of the DSR algorithm is shown below.



```
Results
Time              Element  Message
2.35611324494524  Node4    I want to send a message (msg15) to node 01
2.35611324494524  Node4    The required path wasn't found in the cache. Sending RREQ packets to neighbors for pathfinding
2.37611324494524  Node2    Received RREQ packet: [916;04;01;(empty)]
2.37611324494524  Node2    The packet is not for me. I'm sending RREQ packet to neighbors for pathfinding
2.37611324494524  Node5    Received RREQ packet: [916;04;01;(empty)]
2.37611324494524  Node5    The packet is not for me. I'm sending RREQ packet to neighbors for pathfinding
2.48424017451901  Node2    I want to send a message (msg1) to node 03
2.48424017451901  Node2    The required path wasn't found in the cache. Sending RREQ packets to neighbors for pathfinding
```

Fig.9. The results of smart agents functioning

V. Conclusion

The paper presents linguistic and software tools that can be used for agent-based simulation of routing algorithms in Ad-hoc networks. Syntax constructions for the implementation of reasoning (intelligent) agents are demonstrated and the results of a simulation experiment are presented. The rule-based model of knowledge is used for the implementation of intelligent agents.

Acknowkedgement

The reported study was funded by RFBR and the Krasnodar Region Administration, project number 19-47-230003.

References

[1] (2021) The OMNeT++ Community Site. [Online] Available: http://www.omnetpp.org/

[2] (2021) NS-2 Community Site [Online] Available: http://www.isi.edu /nsnam /ns /

[3] B.Mishra, D.Garg, P.Narang and V.Mishra. "Drone-surveillance for search and rescue in natural disaster" *Computer Communications,* vol. 156, pp. 1-10, 2020.

[4] B.Alzahrani, O.S.Oubbati, A.Barnawi, M.Atiquzzaman and D.Alghazzawi "UAV assistance paradigm: State-of-the-art in applications and challenges" *Journal of Network and Computer Applications,* vol. 166, article 102706, 2020.

[5] R.Sharma, D.K.Lobiyal. "Proficiency Analysis of AODV, DSR and TORA Ad-hoc Routing Protocols for Energy Holes Problem in Wireless Sensor Networks" *Procedia Computer Science,* vol. 57, pp. 1057-1066, 2015.

[6] A.I.Mikov, E.B.Zamyatina and R.A.Mikheev "Towards the Flexibility of Software for Computer Network Simulation", *Proceedings of the 18th International Conference on Computers,* vol. 1-2, pp. 391-397, 2014.