A Formal Model for the Simulation of Mobile Networks

Emanuele Covino

Dipartimento di Informatica
Universitá degli Studi di Bari Aldo Moro
Bari, Italy
emanuele.covino@uniba.it

***Abstract*—We introduce MOTION (MOdeling and simulaTIng mObile ad-hoc Networks), a Java tool that simulates a well known protocol for mobile networks (the Ad-hoc On-demand Distance Vector - AODV); its definition is based on the Abstract State Machine formal model used within the framework ASMETA (ASM mETAmodeling). Morover, we suggest that some protocols for mobile networks could be used to provide a formal definition of social structures and to analyze the related properties.**

***Index Terms*—AODV, Abstract State Machines, Mobile ad-hoc networks, Mobile computing, Social network analysis.**

I. Introduction

Wireless communication among both stationary and mobile devices in absence of physical infrastructure can be established and performed by means of the Mobile Ad-hoc NETwork technology (MANET) [1] [22] [33]. While stationary devices cannot change their location in the network, mobile devices are free to move randomly, entering or leaving the network and changing their relative positions. Each device can broadcast messages inside its radio range only, implying that, outside this area, communication is possible by means of some sort of cooperation among intermediate devices, exclusively. Thus, a communication protocol capable of handling this lack of predictable topology is needed. One of the most popular routing protocols for MANET's is the Ad-hoc On-demand Distance Vector (AODV) [32], together with several variants introduced in order to reduce communication failures due to topology changes. For example, Reverse-AODV (R-AODV) [6] [25] builds all possible routes between source and destination devices: when the primary route fails (typically the shortest one), communication is still provided by the alternative routes. More recently, variants have been proposed to cope with congestion issues [12] [24] and to improve the security on communications, using cryptography to secure data packets during their transmission (Secure-AODV) [41], and adopting the so-called *trust methods*, in which nodes are part of the communication if and only if they are considered trustworthy (Trusted-AODV) [12] [26]. This research area is receiving more attention in the last few years, in the context of smart mobile computing, cloud computing and Cyber Physical Systems [15] [31].

MANET's technology raises several problems related to the analysis of performance, synchronization and concurrency of the network. Moreover, the request of computing services characterized by high quality levels, broad and continuous availability, and inter-operability over heterogeneous platforms, increases the complexity of the systems' architecture. Therefore, it is quite important to be able to verify qualities like responsiveness, robustness, correctness and performance, starting from the early stages of the system's development. In order to do this, many studies are executed with the support of simulators [3] [36] [39]. They can be used to measure and to evaluate performances and to compare different solutions, implementing the network at a low abstraction level but, by their intrinsic nature, they cannot support proofs of correctness, synchronization and deadlock properties, and they cannot model MANET's with a higher abstraction level of specification. To overcome this limitations, formal methods are used to create a model the system. For instance, the process-calculus [35], the Calculus of Mobile Ad Hoc Networks (CMN) [28], and the Algebra for Wireless Networks (AWN) [14] capture essential characteristic of nodes, such as mobility or packets broadcasting. Petri nets have been employed to study the modeling and verification of routing protocols [40], and the evaluation of protocols performances [13].

This kind of state-based models provide a suitable way of representing algorithms, and they are typically equipped with tools (such as CPN Tools [23]) that allow to simulate the algorithms, directly. However, they lack expressiveness, because they only show a single level of abstraction, and they do not provide simple ways for refinements of the executable code. These characteristics are intrinsic in the Abstract State Machine model (ASM) that provides a way to describe algorithms in a simple abstract pseudo-code, which can be translated into a high-level programming language source code [5] [17]. Formal methods are satisfactory for reasoning about properties of the system they describe, but they rarely are useful for studying performance results [8].

In this paper, we use the ASM formalism to define a MANET and to simulate its behaviour; this is achieved by introducing MOTION (MOdeling and simulaTIng mObile ad-hoc Networks), a tool operating within the framework ASMETA (ASM mETAmodeling) [2] [16]. In particular, we adopt the AODV protocol to manage the evolution of the network and to show the behaviour of the application. In Section II, we

Copyright (c) IARIA, 2023. ISBN: 978-1-68558-050-6

recall concepts and definitions of mobile ad-hoc networks and of the specific protocol adopted in order to capture the dynamic behavior of nodes in the network. In Section III, we recall the basics about Abstract State Machine's [4] [5]. We will use this formalism to define and study properties of the network. In Section IV, we outline the definition and behaviour of MOTION, implementing the previous protocol by means of the ASM's formalism. In Section V, we discuss how the mobile networks' model could be used to represent social groups and to study the related interactions (for instance, those occurring within social networks). Conclusions and future work can be found in Section VI.

II. Mobile Ad-hoc NETworks and a routing protocol

Networks of mobile nodes, usually connected by means of a wireless communication system, have been dubbed MANET. Each node of the network can be considered as an autonomous agent that re-arranges its position without conforming to a fixed topology. During its lifetime it can enter or leave the network, and it can change its position, continuously; this means that routes connecting the nodes can rapidly change, because of their mobility and of the limited range of transmission. When a piece of information has to find his path from a source node towards a destination, a routing protocol is needed. In general, a routing protocol specifies how nodes communicate with each other in order to distribute the information within the network; routing algorithms determine this choice, according to some specific principle, and they are able to adjust the route when changes occur, such as disabled or partially available connections, loops, obstructions, or starvation.

Several routing protocols have been proposed; among them, the *Ad-hoc On-demand Distance Vector* (AODV) [32] is one of the most popular (indeed, a number of simulation studies are dealing with it, representing a reliable baseline for comparison to the results of simulations executed with MOTION). It is a reactive protocol that combines two mechanisms, the *route discovery* and the *route maintenance*, in order to store some knowledge about the routes into *routing tables*. Each node has its own routing table that consists of a list of all the discovered (and still valid) routes towards other nodes in the network; in particular, the routing table entry of the node i concerning a node j includes the *address* of j, the last known *sequence number* of j, the *hop count* field (a measure of the distance between i and j), and the *next hop* field (identifying the next node in the route between i and j). The sequence number is an increasing integer maintained by each node that expresses the freshness of the information about the respective node. When an *initiator node* wants to start a communication session towards the *destination node*, it checks if a route is currently stored in its routing table. If this happens, the communication can start. If there aren't any routes to the destination, the initiator sends a *route request* (RREQ) to all its neighbors. This message includes the initiator address, the destination address, the sequence number of the destination (i.e., the most recent information about the destination), and the hop count, initially set to 0, and increased by each intermediate node. When an intermediate node N receives an RREQ, it creates a routing table entry for the initiator, or, if the entry already exists, it updates its sequence number and next hop. Then, the process is iterated: N checks if it knows a route to the destination with corresponding sequence number greater than the number contained into the RREQ (this means that its knowledge about the route is more recent). If so, N sends back to the initiator a *route reply* (RREP); otherwise, N updates the hop count field and broadcasts once more the RREQ to all its neighbors. The process ends successfully when a route to the destination is found. While the RREP travels towards the initiator, the routing tables of the traversed nodes are updated, creating an entry for the destination, when needed. Once the initiator receives back the RREP, the communication can start. The mobile nature of the nodes can create new routes or break some of them, because new links are established betweens pairs of nodes or because one or more links are no more available; when this happens, a route maintenance is executed in order to notify the error and to invalidate the corresponding routes, propagating a *route error* (RERR) into the network.

III. Abstract State Machines

An ASM [5] M is a tuple (Σ, S, R, P_M). Σ is a *signature*, that is, a finite collection of names of total functions; each function has arity n, and the special value *undef* belongs to the range (*undef* represents an undetermined object, the default value). Relations are expressed as particular functions that always evaluate to *true*, *false* or *undef*.

S is a finite set of *abstract states*. The concept of abstract state extends the usual notion of state occurring in finite state machines: it is an algebra over the signature Σ, i.e., a non-empty set of objects together with interpretations of the functions in Σ. Pairs of function names, together with values for their arguments, are called *locations*: they are the abstraction of the notion of memory unit. Since a state can be viewed as a function that maps locations to their values, the current configuration of locations, together with their values, determines the current state of the ASM.

R is a finite set of *rule declarations* built starting from the *transition rules* skip, update ($f(t_1, t_2, \ldots, t_n) := t$), conditional (**if** ϕ **then** P **else** Q), let (**let** $x = t$ **in** P), choose (**choose** x **with** ϕ **do** P), sequence (P **seq** Q), call ($r(t_1, \ldots, t_n)$), block (P **par** Q) (see [5] for their operational semantics). The rules transform the states of the machine, and they reflect the notion of transitions occurring in traditional transition systems. A distinguished rule P_M, called the *main rule* of the machine, represents the starting point of the computation.

A *move* of a ASM, in a given state, consists of the simultaneous execution of all the rules whose conditions evaluates to true in that state. Since different updates could affect the same location, it is necessary to impose a consistency requirement: a set of updates is said to be *consistent* if it contains no pairs of updates referring to the same location. Therefore, if the updates are consistent, the result of a move is the transition of

Copyright (c) IARIA, 2023. ISBN: 978-1-68558-050-6

the machine from the current state to another; otherwise, the computation doesn't produce a next state. A *run* is a (possibly infinite) sequence of moves: they are iterated until no more rules are applicable.

The aforementioned notions refer to the *basic* ASMs. However, there exist some generalisations (e.g., Parallel ASMs and Distributed ASMs) [17]. Parallel ASMs are basic ASMs enriched with the rule forall x with ϕ do P, to express the simultaneous execution of the same ASM P over x satisfying the condition ϕ. A Distributed ASM is intended as a finite number of independent agents, each one executing its own underlying ASM: it is capable of capturing the formalization of multiple agents acting in a distributed environment. A run, which is defined for sequential systems as a sequence of computation steps of a single agent, is defined as a partial order of moves of finitely many agents, such that the three conditions of co-finiteness, sequentiality of single agents, and coherence are satisfied. Roughly speaking, a global state corresponds to the union of the signatures of each ASM together with interpretations of their functions.

IV. Defining MANET's by means of ASM

In [9], we have given a description of a MANET's behaviour based on the parallel ASM model and we have introduced a preliminary version of MOTION that allows to define its parameters (such as mobility and level of activity of a node), to run the network, and to collect the output data of the simulation. In this paper, we provide a refinement that allows the user to visualize the dynamic evolution of the network, step by step: the mobility of nodes within the network, the path from a source to a destination and the overall evolution of the network can be monitored and studied. The complete package can be found in [21].

MOTION is developed within the ASMETA framework [16]; the behaviour of the network is modelled using the AsmetaL language, and then the model is executed by the AsmetaS simulator. The executions of MOTION and ASMETA are interleaved: first, MOTION captures the parameters of the network (number of nodes and their level of mobility, for instance) and includes them into an AsmetaL file; then, it runs AsmetaS according to those parameters. AsmetaS executes an ASM move, simulating the behavior of the protocol over the current network's configuration. The control goes back to MOTION at the end of each move: the information related to the move (such as the new positions of the nodes, the sent/received requests, the relations among the nodes) are recorded and, in this new version, the current topology of the network is visualised (showing the successful communication attempts between pairs of nodes, the connections established, and the failed attempts). Then, MOTION invokes AsmetaS for the next move. At the end of the simulation, MOTION reads the final log file, parses it, and stores the collected results in a csv file. Note that these interleaved calls require a considerable amount of interaction work; this is done in order to collect the information about the evolution of the network step by step, and to use it for the analysis of the behaviour of the network itself.

In more details, MOTION expresses the network topology by means of an *adjacency matrix C*, such that $c_{ij} = 1$ if i and j are neighbors, 0 otherwise, for each pair of nodes i and j. The mobility of nodes is implemented by updating the adjacency matrix at every step of the simulation; each c_{ij} is randomly set to 0 or 1, according to a mobility parameter defined by the user. The new values of the matrix are used to execute the next ASM move, accordingly. The relations among nodes are expressed by means of predicates, as expected: for instance, the reachability between two agents a_i and a_j is expressed by the predicate isLinked(a_i, a_j), which evaluates to *true* if there exists a coherent path from a_i to a_j, to *false* otherwise; the predicate knowsActiveRouteTo(a_j, a_j) states that a_i has an active path leading to a_j into its routing table.

The AODV routing protocol has been formally modeled through ASMs in [4], for the first time. MOTION redefines the protocol by means of new predicates and rules, also adding a parameter *Timeout*, the waiting time for the route reply, to avoid infinite loops when searching for a route. Each node of the network represents a device or an agent. In what follows, we show some of the high-level rules of MOTION (notice the use of **forall** in order to run AODVSPEC on every node in the network, and to look for a route from a given source a to the remaining nodes dest).

```
MAIN RULE AODV =
      forall a ∈ Nodes do AODVSPEC(a)


AODVSPEC(a)=
   forall dest ∈ Nodes with dest ≠ a do
      if WaitingForRouteTo(a, dest) then
         if Timeout(a, dest) > 0 then
            Timeout(a, dest) := Timeout(a, dest)-1
         else
            par
               WaitingForRouteTo(a, dest) := false
               ca-fail(a, dest) := ca-fail(a,dest)+1
            endpar
         endif
   if WishToInitiate(a) then PREPARECOMM(a)
   if not Empty(Message) then ROUTER
```

WaitingForRouteTo expresses that the discovery process previously started is still running. In this case, if the waiting time for RREP is not expired (i.e., Timeout() > 0), the time-counter is decreased; otherwise, the search for the route is ended. If *WishToInitiate* evaluates to true (depending on a *initiator probability* parameter), the node wants to start a communication, and the following rule PREPARECOMM is called.

```
PREPARECOMM(a) =
  forall dest ∈ Nodes with dest ≠ a do
    choose wantsToCommWith ∈ Boolean with true do
```

```
if wantsToCommWith then
  par
    if not waitingForRouteTo(a,dest) then
      ca-tot(a, dest) := ca-tot(a, dest) + 1
    endif
    if knowsActiveRouteTo(a,dest) then
      par
        StartCommunicationWith(dest)
        waitingForRouteTo(a, dest) := false
      endpar
    else
      if not waitingForRouteTo(a, dest) then
        par
          GenerateRouteReq(dest)
          WaitingForRouteTo(a, dest) := true
          Timeout(a,dest) := Timeout
        endpar
      endif
    endif
  endpar
endif
```

Finally, if the node has received a message (either RREQ, RREP or RERR), ROUTER is called, with

```
ROUTER = ProcessRouteReq;
         ProcessRouteRep;
         ProcessRouteErr
```

where each sub-rule expresses the behavior of the node, depending on the type of the message received. Thanks to this formalization, some properties have been proven in the past, such as the starvation freeness for the protocol, the properness of the message received back by the initiator of any communication, and the capability to intercept black holes into the network.

An actual simulation in MOTION is performed in a number of sessions established by the user (10 sessions, Figure 1), each of which has a duration (50 moves, Figure 1); during each session, the network has a number of agents (hosts) defined by the user. Each agent tries to initiate a communication towards a destination: the probability that one of them acts as an initiator is defined by setting the parameter *Initiator Probability* (10 per cent, Figure 1). Thanks to the intrinsic parallelism in the execution of the ASM's rules, more attempts can be executed simultaneously. A communication attempt is considered successful if the initiator receives an RREP within the waiting time expressed by the parameter *Timeout*; otherwise, the attempt is considered failed.

In MOTION, agents' mobility is defined by the user by means of two parameters, namely *Initial connectivity* and *Mobility level*. The former defines the initial topology of the MANET: it expresses the probability that each agent is directly linked to any other agent. During the simulation, the mobility of agents is expressed by the random re-definition of the values of the *adjacency matrix C*. More precisely, for each pair of agents (a_i, a_j), and for each move of the ASM, the values of C are changed with a probability expressed by *Mobility level*.

The new version of MOTION starts from an interface that allows to set the parameters of the network (Figure 2); in this case, six agents populate the network, with a high value of initial connectivity and a low level of mobility. The chance that an agent starts a communication is set to 20 per cent. When the simulation is started, some new dynamic windows are visualised, in contrast with the previous version of the tool. For instance, a step of the network evolution can be seen in Figure 3. The window *mobility model* represents the connectivity matrix, that is, the existing direct connections among nodes; because of the high initial connectivity, we can find a big number of *successful connections* and no *failed connections*. After several moves, Figure 4 shows a new mobility model, and a new set of successful or failed connections.

V. Social networks analysis

Social structures can be investigated by means of methods and tools of social network analysis. A model often used to represent these structures is a graph or network, that is, a collection of nodes connected by arcs; the former are associated with people or agents, while the latter represent any kind of relation, interaction or influence between pairs (or groups) of agents [30]. This idea has been applied in a large number of studies, about social media networks [18] [20], information circulation [19] [29], business networks, knowledge networks [7] [11]. In particular, social network analysis is a key technique in modern sociology, demography, communication studies, market economy, sociolinguistic, cooperative learning, being able to represent data by means of a simple data structure, a graph, and to analyze the intrinsic interactions using the standard methods and measures provided by mathematics and computer science [38]. The interest of scientists is surely driven by the availability of the so-called big data; between 1990 and 2005, the new (virtually) unbounded computational power has been applied to the concept of self-organizing systems, providing the definition of models and simulations of a big number of social activities. In the mid 1990s, physicist and mathematicians started to analyze big data from financial markets, resulting in the development of econophysics [27]; in the 2000s, the focus shifted on big data generated by the Internet and the social networks, looking for characteristic patterns that exists in social interactions, no matter the technology, and revitalizing the research in sociophysics [34] and in computational social sciences. Many studies are executed with the support of simulators that are suitable to compare different social structures and several scenarios, according to the parameters of the network.

In general, networks used to represent social interactions are static, meaning that the location of nodes and the related ties don't change as time goes by; every change that may happen in the social group is not captured by this model. Aside static networks, mobile networks exist: they have a flexible structure, and their topology changes dynamically, given that nodes can join or leave the network during their lifetime, that

communication among them depends on the availability of a connection, and that connections can have different strength. This reflects the dynamic nature of ties that exists between agents in a social group. Computer science provides methods to define and represent these kind of networks, together with algorithms that allow to broadcast a message from a source to a destination, mimicking the spread of information, opinions, or consensus into the group. In order to do this, agents should behave according to a cooperation protocol. We suggest that the MANET models, as well as other models of mobile networks, could be used to represent a social group and to study the related interactions [10]. MOTION could be used by social scientists to represent and study social interactions. For instance, a high value of the *initial connectivity* parameter, together with a low level of *mobility*, represent strong ties within a very cohesive group, meaning that the members of the group do not change their opinion or do not end a relation easily. On the contrary, a high mobility means that the group is prone to change opinions very easily. The *initiator probability* measures how much a member of a social group is inclined to spread information inside the network. It appears that the properties of a MANET match the properties that can be found in a social group, like starvation of information, fake information spreading, popularity of opinions, and so on. One could follow the propagation of a message (an opinion, an influence) inside the social group that is represented by the network, and to study how this is affected by the mobility of the agents or by the strength of the ties inside the group itself.

VI. Conclusions and future work

MANET is a technology used to perform wireless communications among mobile devices in absence of physical infrastructure. It is widely used in the context of smart mobile computing, cloud computing and Cyber Physical Systems. Several routing protocols have been developed, and problems have been raised about the measurement of performances, and also about the formal analysis of qualities like responsiveness, robustness, correctness. In order to address these problems, both simulators and formal description methods are needed. The former allow us to measure performances through direct simulation, but they aren't suitable to investigate the properties of the networks. This can be achieved when using formal methods, but they can hardly be used to measure performance. In this paper, we have introduced MOTION, a Java application in which MANET's are modeled as an Abstract State Machine by means of the AsmetaL representation. This representation can be used to prove formal properties of the network, as well as can be simulated by means of the simulation engine AsmetaS. MOTION can collect the results of this simulation that can be used for performances' analysis. We have validated MOTION on the Ad-hoc On-demand Distance Vector protocol.

Several variants of routing protocols for mobile networks have been proposed in the past; among them, the *NACK-based Ad-hoc On-demand Distance Vector* (N-AODV), that improves the awareness that each host has about the network topology, and the *Blackhole-free N-AODV* (BN-AODV), that detects the presence of malicious nodes leading to a blackhole attack.

One of the disadvantages of the AODV protocol is the poor knowledge that each node has about the network topology. In fact, each node n is aware of the existence of a node m only when n receives an RREQ, either originated by, or directed to m. In order to overcome this limitation, the NACK-based AODV routing protocol has been proposed and modeled by means of a Distributed ASM. A *Not ACKnowledgment* (NACK) control packet is added in the route discovery phase. Whenever an RREQ originated by n and directed to m is received by the node p that doesn't know anything about m, p unicasts the NACK to n, with the purpose to state the ignorance of p about m. In this way, n (as well as all the nodes in the path to it) receives fresh information about the existence and the relative position of p. Therefore, on receiving the NACK, all the nodes in the path to p add an entry in their respective routing tables, or update the pre-existing entry. N-AODV has been experimentally validated through simulations, showing its efficiency and effectiveness: the nodes in the network actually improve their knowledge about the other nodes and, in the long run, the number of RREQ decreases, with respect to the AODV protocol.

All routing protocols assume the trustworthiness of each node; this implies that MANET's are prone to the *black hole attack* [37]. In AODV and N-AODV a black hole node produces fakes RREPs in which the sequence number is as great as possible; the initiator establishes the communication with the malicious node and the latter can misuse or discard the received information. The black hole can be supported by one or more *colluders* that confirm the trustworthiness of the fake RREP. The Black hole-free N-AODV protocol allows the honest nodes to intercept the black holes and the colluders, thanks to two control packets: each intermediate node n receiving an RREP must verify the trustworthiness of the nodes in the path followed by the RREP; to do this, n produces a *challenge packet* (CHL) for the destination node, and only the latter can produce the correct *response packet* (RES). If n receives RES, it sends the RREP, otherwise the next node towards the destination is a possible black hole.

We are currently working on the final definition of the ASM for the N-AODV and the BN-AODV protocols, together with the extension of MOTION to those protocols. Moreover, a complexity analysis of the network's protocols and the related algorithms could be performed; a change of the structure that represents the connectivity among the nodes (from adjacency matrix to adjacency list, for instance), could lead to a dramatic improvement of the resource-consumption during the simulation of the behaviour of the network.

Copyright (c) IARIA, 2023. ISBN: 978-1-68558-050-6

REFERENCES

[1] D. P. Agrawal and Q.-A. Zeng, Introduction to wireless and mobile systems. Cengage learning - Fourth Edition, Boston, 2016.

[2] P. Arcaini, A. Gargantini, E. Riccobene, and P. Scandurra, "A model-driven process for engineering a toolset for a formal method," Software: Practice and Experience, vol. 41(2), pp. 155–166, 2011.

[3] S. Basagni, M. Mastrogiovanni, A. Panconesi, and C. Petrioli, "Localized protocols for ad hoc clustering and backbone formation: A performance comparison," IEEE Trans. Parallel Distrib. Syst., vol. 17(4), pp. 292–306, 2006, doi:10.1109/TPDS.2006.52, URL https://doi.org/10.1109/TPDS.2006.52

[4] E. Börger and A. Raschke, Modeling Companion for Software Practitioners. Springer, 2018. doi:10.1007/978-3-662-56641-1. URL https://doi.org/10.1007/978-3-662-56641-1

[5] E. Börger and R. Stärk, Abstract State Machines: A Method for High-Level System Design and Analysis. Springer Verlag, Berlin, 2003.

[6] L. Bononi, G. D'Angelo, and L. Donatiello, "Hla-based adaptive distributed simulation of wireless mobile systems," Proceedings of the seventeenth workshop on Parallel and distributed simulation, p. 40, IEEE Computer Society, 2003.

[7] J. Brennecke and O. Rank, "The firm's knowledge network and the transfer of advice among corporate inventors — A multilevel network study," Research Policy, 46(4), pp. 768-783, 2017.

[8] R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola, "Self-adaptive software needs quantitative verification at runtime," Communications of the ACM, vol. 55(9), pp. 69–77, 2012.

[9] E. Covino and G. Pani, "Analysis of Mobile Networks' Protocols Based on Abstract State Machines," in A. Raschke et al. (Eds.): Logic, computation and rigorous methods, LNCS 12750, pp. 187-198, 2021.

[10] E. Covino and G. Pani, "Analysis of a Formal Model for Social Groups," ICOMP'22 - The 23rd Int'l Conf on Internet Computing and Internet of Things, July 25th-28th, 2022, Las Vegas, USA.

[11] A. D'Andrea, F. Ferri, and P. Grifoni, "An Overview of Methods for Virtual Social Network Analysis," in Abraham, Ajith (Ed.), Computational Social Network Analysis: Trends, Tools and Research Advances, Springer, pp. 3-25, 2009.

[12] N. Das, S. K. Bisoy, and S. Tanty, "Performance analysis of TCP variants using routing protocols of manet in grid topology," Cognitive Informatics and Soft Computing, pp. 239–245, Springer, 2019.

[13] F. Erbas, K. Kyamakya, and K. Jobmann, "Modelling and performance analysis of a novel position-based reliable unicast and multicast routing method using coloured Petri nets," 2003 IEEE 58th Vehicular Technology Conference, VTC 2003-Fall, Vol. 5, pp. 3099–3104, IEEE, 2003.

[14] A. Fehnker et al., "A process algebra for wireless mesh networks," European Symposium on Programming, pp. 295–315, Springer, 2012.

[15] A. Garcia-Santiago, J. Castaneda-Camacho, J. F. Guerrero-Castellanos, G. Mino-Aguilar, and V. Y. Ponce-Hinestroza, "Simulation platform for a VANET using the truetime toolbox: Further result toward cyber-physical vehicle systems," IEEE 88th Vehicular Technology Conference (VTC-Fall), IEEE, pp. 1–5, 2018.

[16] A. Gargantini, E. Riccobene, and P. Scandurra, "A metamodel-based language and a simulation engine for abstract state machines," J. UCS, vol. 14(12), pp. 1949–1983, 2008. doi:10.3217/jucs-014-12-1949. URL https://doi.org/10.3217/jucs-014-12-1949

[17] U. Glässer, Y. Gurevich, and M. Veanes, "Abstract communication model for distributed systems," IEEE Trans. Software Eng., 30(7), pp. 458–472, 2004. doi:10.1109/TSE.2004.25. URL https://doi.org/10.1109/TSE.2004.25

[18] M. Grandjean, "A social network analysis of Twitter: Mapping the digital humanities community," Cogent Arts and Humanities, 3(1), 2016.

[19] M. Grandjean, "Analisi e visualizzazioni delle reti in storia. L'esempio della cooperazione intellettuale della Società delle Nazioni," Memoria e Ricerca, 2, pp. 371–393, 2017.

[20] L. Hagen, T. Keller, S. Neely, N. DePaula, and C. Robert-Cooperman, "Crisis Communications in the Age of Social Media: A Network Analysis of Zika-Related Tweets," Social Science Computer Review, 36(5), pp. 523-541, 2018.

[21] URL https://github.com/Angelo997/VisualMotion.git.

[22] M. Ilyas (Ed.), The Handbook of Ad Hoc Wireless Networks, (1st ed.). CRC Press, 2002.

[23] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured Petri nets and CPN tools for modelling and validation of concurrent systems," International Journal on Software Tools for Technology Transfer, vol. 9(3-4), pp. 213–254, 2007.

[24] N. Kaur and R. Singhai, "Analysis of traffic impact on proposed congestion control scheme in AODV," Wireless Personal Communications, pp. 1–24, 2019.

[25] C. Kim, E. Talipov, and B. Ahn, "A reverse AODV routing protocol in ad hoc mobile networks," International Conference on Embedded and Ubiquitous Computing, pp. 522–531, Springer, 2006.

[26] X. Li, M. R. Lyu, and J. Liu, "A trust model based routing protocol for secure ad hoc networks," in 2004 IEEE Aerospace Conference Proceedings, Vol. 2, pp. 1286–1295, IEEE, 2004.

[27] R. Mantegna and H. Stanley, Introduction to Econophysics: Correlations and Complexity in Finance. Cambridge University Press, 1999.

[28] M. Merro, "An observational theory for mobile ad hoc networks," Information and Computation, vol. 207(2), pp. 194–208, 2009.

[29] H. R. Nasrinpour, M. R. Friesen, and R. D. McLeod, "An Agent-Based Model of Message Propagation in the Facebook Electronic Social Network," arXiv:1611.07454, 2016.

[30] E. Otte and R. Rousseau, "Social network analysis: a powerful strategy, also for the information sciences," Journal of Information Science, 28(6), pp. 441–453, 2002.

[31] A. P. Pandian, J. I.-Z. Chen, and Z. A. Baig, "Sustainable mobile networks and its applications," Mobile networks and application, vol. 24(2), pp. 295–297, 2019.

[32] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc on-demand distance vector (AODV) routing," RFC 3561 (2003), pp. 1–37, doi:10.17487/RFC3561. URL https://doi.org/10.17487/RFC3561.

[33] R. R. Roy, Handbook of Mobile Ad Hoc Networks for Mobility Models. Springer, 2011.

[34] F. Schweitzer, "Sociophysics," Physics Today, 71(2), p. 40, 2018.

[35] A. Singh, C. Ramakrishnan, and S. A. Smolka, "A process calculus for mobile ad-hoc networks," Science of Computer Programming, vol.75(6), pp. 440–469, 2010.

[36] D. A. Tran and H. Raghavendra, "Congestion adaptive routing in mobile ad-hoc networks," IEEE Trans. Parallel Distrib. Syst., vol. 17(11), pp. 1294–1305, 2006. doi:10.1109/TPDS.2006.15. URL https://doi.org/10.1109/TPDS.2006.151.

[37] F.-H. Tseng, L.-D. Chou, and H.-C. Chao, "A survey of black hole attacks in wireless mobile ad hoc networks," Human-centric computing and information sciences, 1,4, 2011.

[38] S. Wasserman and K. Faust, Social Networks Analysis: Methods and Applications. Cambridge University Press, 1994.

[39] J. Wu and F. Dai, "Mobility-sensitive topology control in mobile ad hoc networks," IEEE Trans. Parallel Distrib. Syst., vol. 17(6), pp. 522–535, doi:10.1109/TPDS.2006.73, URL https://doi.org/10.1109/TPDS.2006.73.

[40] C. Xiong, T. Murata, and J. Leigh, "An approach for verifying routing protocols in mobile ad hoc networks using Petri nets," in Proceedings of the IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication, Vol. 2, pp. 537–540, IEEE, 2004.

[41] M. G. Zapata, "Secure ad hoc on-demand distance vector routing," ACM SIGMOBILE Mobile Computing and Communications Review, 6(3), pp. 106–107, 2002.

Copyright (c) IARIA, 2023. ISBN: 978-1-68558-050-6

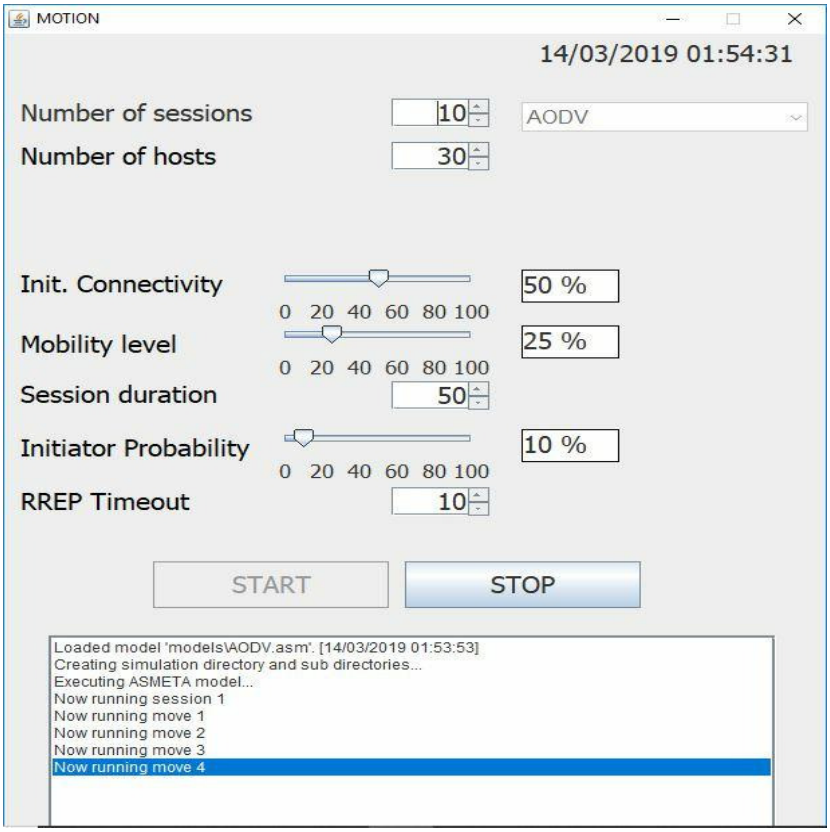

Fig. 1. MOTION's user interface for AODV protocol

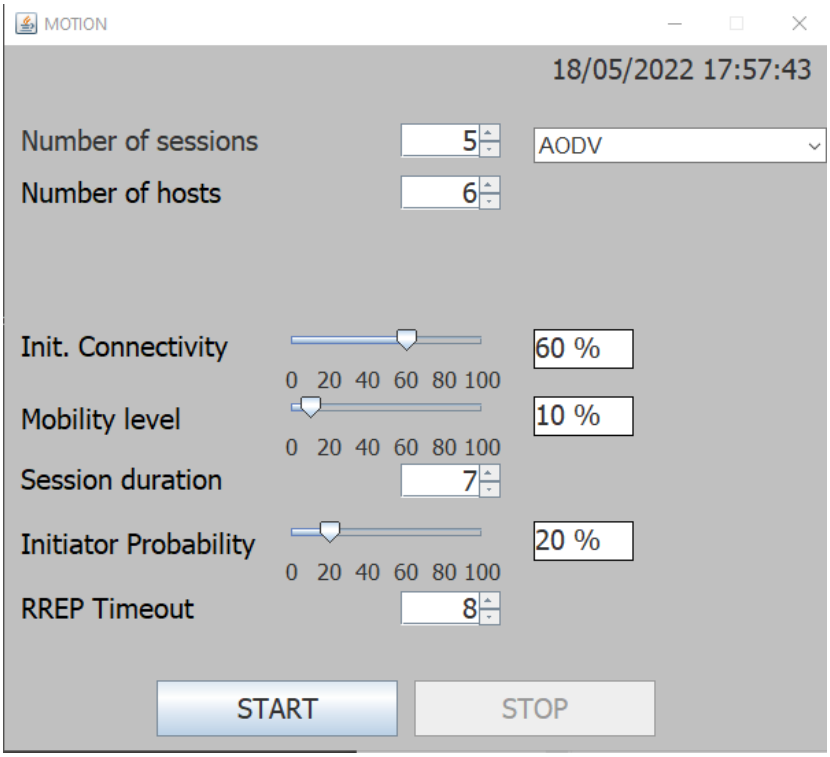

Fig. 2. MOTION's new user interface

Copyright (c) IARIA, 2023. ISBN: 978-1-68558-050-6

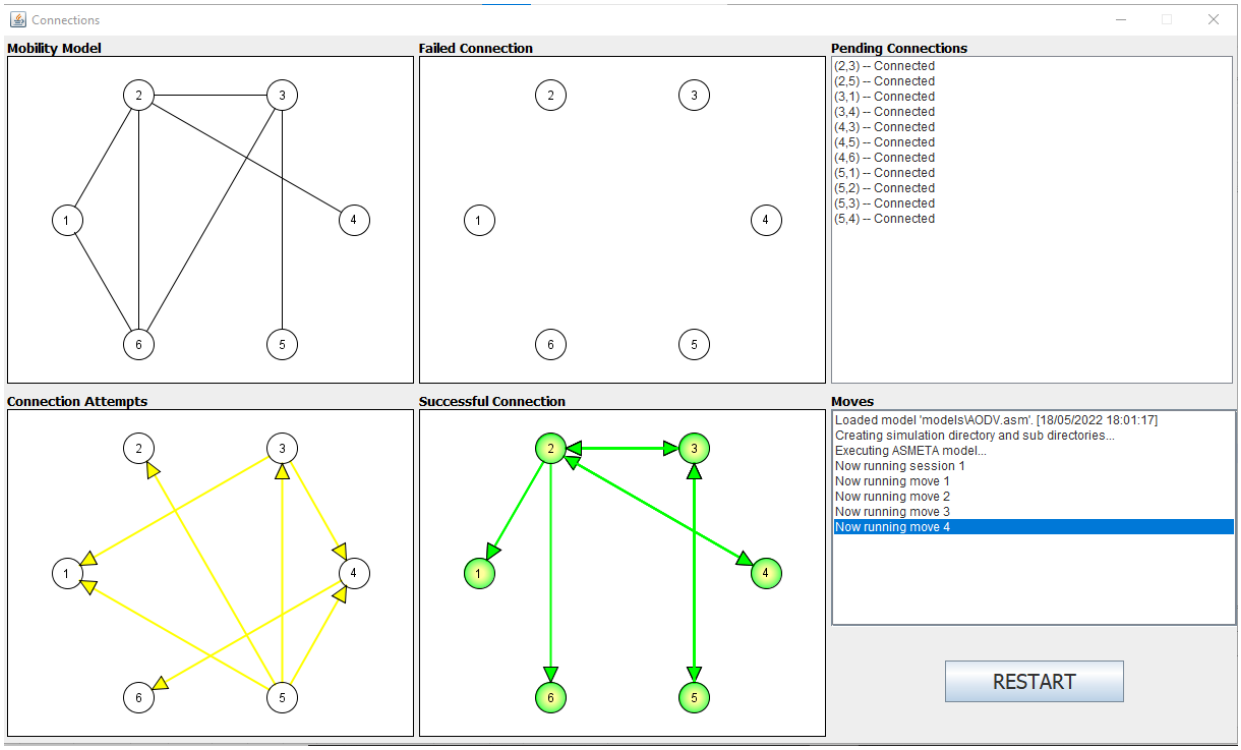

Fig. 3. Evolution of the network

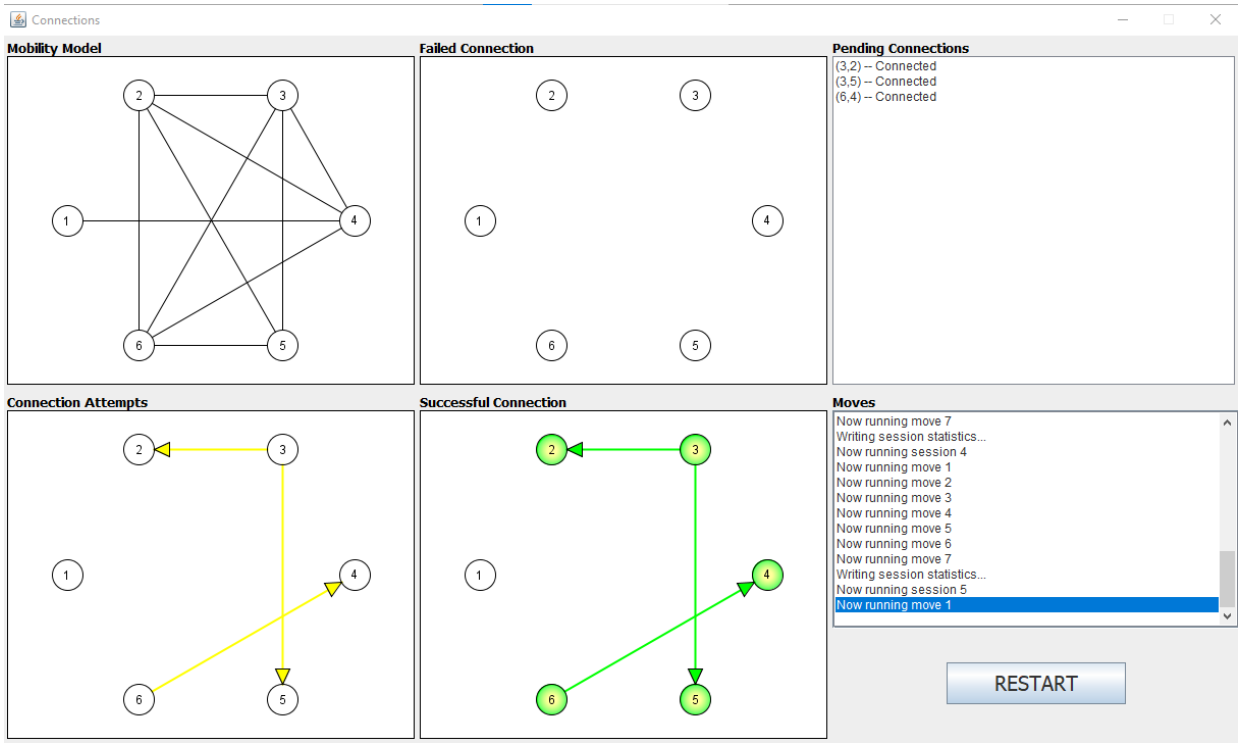

Fig. 4. Evolution of the network, after several steps

Copyright (c) IARIA, 2023. ISBN: 978-1-68558-050-6