Analysis of Mobile Networks' Protocols Based on Abstract State Machine

Emanuele Covino and Giovanni Pani

Dipartimento di Informatica
Universitá di Bari, Italy
emanuele.covino@uniba.it, giovanni.pani@uniba.it

Abstract. We define MOTION (MOdeling and simulaTIng mObile ad-hoc Networks), a Java application based on the framework ASMETA (ASM mETAmodeling), that uses the ASM (Abstract State Machine) formalism to model and simulate mobile networks. In particular, the AODV (Ad-hoc On-demand Distance Vector) protocol is used to show the behaviour of the application.

Keywords: Abstract State Machines; mobile ad-hoc networks.

1 Introduction

Mobile Ad-hoc NETwork (MANET) is a technology used to establish and to perform wireless communication among both stationary and mobile devices in absence of physical infrastructure [1]. While stationary devices cannot change their physical location, mobile devices are free to move randomly: they can enter or leave the network and change their relative positions. Thus, the network lacks a predictable topology. Each device is able to broadcast messages inside its radio range only; outside this area, communication is possible only by means of co-operation between intermediate devices. They can act as initiator, intermediate and destination of a communication. This research area is receiving attention in the last few years, in the context of smart mobile computing, cloud computing and Cyber Physical Systems ([21] and [13]).

One of the most popular routing protocols for MANETs is the Ad-hoc On-demand Distance Vector (AODV, [22]), and several variants have been introduced in order to reduce communication failures due to topology changes. For example, Reverse-AODV (R-AODV, [18] and [8]) overcomes this problem by building all possible routes between initiator and destination: in case of failure of the primary route (typically the shortest one), communication is still provided by the alternative routes. More recently, variants have been proposed to cope with congestion issues ([17] and [10]) and to improve the security on communications, using cryptography to secure data packets during their transmission (Secure-AODV, [29]), 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, [19] and [10]).

The technology of Mobile Ad-hoc NETwork (MANET) raises several problems about 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 important to be able to verify qualities like responsiveness, robustness, correctness and performance, starting from the early stages of the development. To do this, many studies are executed with the support of simulators [3], [27], [25]. They are suitable to evaluate performance and to compare different solutions, implementing the network at a low abstraction level, and considering only a limited range of scenarios. The simulators, by their intrinsic nature, cannot provide specification at higher level, and they cannot support proofs of correctness, of synchronization and of deadlock properties. They measure performances, but they cannot model MANETs with a higher abstraction level of specification.

To do this, formal methods that model the process are needed. For instance, the process-calculus [24], CMN (Calculus of Mobile Ad Hoc Networks, [20]), and AWN (Algebra for Wireless Networks, [12]) 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 [28] and the evaluation of protocols performance [11]. With respect to process calculi, state-based models provide a suitable way of representing algorithms, and they are typically equipped with tools, such as CPN Tools [16], that allow to simulate the algorithms, directly. However, we believe that proposed state-based models lack expressiveness: basically, they provide only a single level of abstraction, and cannot support refinements to executable code. Instead, this characteristic is intrinsic in the ASM model. Even if formal methods are satisfactory for reasoning about correctness properties, they rarely are useful for studying performance properties [9]. Generally speaking, correctness properties are formally proved, while the performance properties are investigated through simulations of the system.

Our aim is to use the ASM formalism to study formal properties, and to use MOTION as a tool for evaluating performance properties. The ASM approach provides a way to describe algorithms in a simple abstract pseudo-code, which can be translated into a high-level programming language source code, as in [7] and in [15].

In Section 2, we recall concepts and definitions related to the Abstract State Machine's model. In Section 3, we describe three mobile network's protocols: AODV (Ad-hoc On-demand Distance Vector), N-AODV (NACK-based AODV), and BN-AODV (Black hole-free N-AODV). In Section 4, we introduce the definition and specific behaviour of MOTION, with respect to the ASM's model of the previous network protocols. Conclusions and future work can be found in Section 5.

2 Abstract State Machines

An Abstract State Machine (ASM, [7]) 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 [7] for their operational semantics). The rules transform the states of the machine, and they reflect the notion of transition 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 execution of all the rules whose conditions are 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 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 (DASMs) [15]. 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.

3 MANET and routing protocols

Mobile Ad-hoc NETworks are networks of autonomous mobile nodes whose topology is not predefined. Each node has a transmission radio range within which it can transmit data to other nodes, directly. Because of the potential movements of the nodes, the routes connecting them can change rapidly.

Several routing protocols have been proposed; among them, the *Ad-hoc On-demand Distance Vector* (AODV) is one of the most popular. Indeed, a large number of simulation studies are dealing with it, representing a reliable baseline for comparison to the results of simulations executed with MOTION. Moreover, we add two variants of AODV: *NACK-based Ad-hoc On-demand Distance Vector* (N-AODV, [4]), that improves the awareness that each host has about the network topology, and *Blackhole-free N-AODV* (BN-AODV, [5]), that detects the presence of malicious nodes leading to a blackhole attack.

3.1 Ad-hoc On-demand Distance Vector (AODV)

This routing protocol has been defined in [22]: it is a reactive protocol that combines two mechanisms, namely the *route discovery* and the *route maintenance*, in order to store some knowledge about the routes into *routing tables*. The routing table associated with each node is a list of all the discovered (and still valid) routes towards other nodes in the network, together with other information. In particular, for the purposes of the present paper, an entry of the routing table of the node i concerning a node j includes: the *address* of j; the last known *sequence number* of j; the *hop count* field, expressing the distance between i and j; and the *next hop* field, identifying the next node in the route to reach j.

The sequence number is an increasing number maintained by each node, that expresses the freshness of the information about the respective node. When an *initiator* wants to start a communication session towards the *destination*, it checks if a route is currently stored in its routing table. If so, the protocol ends and the communication starts. Otherwise, the initiator broadcasts a control packet called *route request* (RREQ) to all its neighbors.

An RREQ packet includes the initiator address and broadcast id, the destination address, the sequence number of the destination (i.e., the latest available information about the destination), and the hop count, initially set to 0, and increased by each intermediate node. The pair <*initiator address*; *broadcast id*> identifies the packet, uniquely; this implies that duplications of RREQs already handled by nodes can be ignored.

When an intermediate node n receives an RREQ, it creates the routing table entry for the initiator, or updates it in the fields related to the sequence number and to the next hop. Then, the process is iterated: n checks if it knows a route to the destination with corresponding sequence number greater than the one contained into the RREQ (this means that its knowledge about the route is more recent). If so, n unicasts a second control packet (the *route reply*, RREP) back to the initiator. Otherwise, n updates the hop count field and broadcasts once more the RREQ to all its neighbors.

The process successfully ends when a route to the destination is found. While the RREP travels towards the initiator, routes are updated inside the routing tables of the traversed nodes, creating an entry for the destination, when needed. Once the initiator receives back the RREP, the communication can start. If the nodes' movements break a link (i.e., a logical link stored in a routing table is no more available), a route maintenance is executed in order to notify the error and to invalidate the corresponding routes: to this end the control packet *route error* (RERR) is used.

3.2 NACK-based AODV (N-AODV)

One of the main 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 improve the network topology awareness, the NACK-based AODV routing protocol has been proposed and modeled by means of a Distributed ASM in [4].

This protocol is a variant of AODV: it adds a *Not ACKnowledgment* (NACK) control packet 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. The purpose of this control packet is 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.

3.3 Black hole-free N-AODV (BN-AODV)

All routing protocols assume the trustworthiness of each node; this implies that MANETS are very prone to the *black hole attack* [26]. In AODV and N-AODV a black hole node produces fakes RREPs, in which the sequence number is as great as possible, so that 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 [5] 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.

4 MOTION

4.1 Development and behavior

As stated before, MOTION (MOdeling and simulaTIng mObile ad-hoc Networks) is a Java application that allows to specify the simulation parameters, to execute the network described, and to collect the output data of the simulation.

To define MOTION, we have used the ASM-based method consisting in development phases, from requirements' specification to implementation. Some environments support this method, and among them the ASMETA (ASM mETAmodeling) framework [2], [14]. This framework is characterized by logical components that capture the requirements by constructing the so-called *ground models*, i.e. representations at high level of abstraction that can be graphically depicted. Starting from ground models, hierarchies of intermediate models can be built by stepwise refinements, leading to executable code: each refinement describes the same system at a finer granularity. The framework supports both verification, through formal proof, and validation, through simulation.

MOTION is developed within the ASMETA framework thanks to the abstract syntax defined in the AsmM metamodel; the behavior of the MANET is modelled using the AsmetaL language, and then the network is executed by the AsmetaS simulator. Since AsmetaS simulates instances of the model expressed by means of the AsmetaL, the information concerning each instance (number of agents and their features, for instance) must be recorded into the AsmetaL file.

The executions of MOTION and ASMETA are interleaved: MOTION provides the user interface and captures the data inserted by the user, representing the parameters of the simulation. MOTION then includes these data into the AsmetaL file, and it runs AsmetaS. AsmetaS executes an ASM move, simulating the behavior of the network protocol over the current data, and it records the values of the locations in a log file, for each state. At the end of each move the control goes back to MOTION: it gets the information about the results of the ASM move, such as the relative position of the hosts, the sent/received packets, and the values of waiting time, and it records them into the AsmetaL file. Then, MOTION invokes AsmetaS for the next move. Even if this interleaved executions requires a good 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 performances and behaviour of the network itself.

At the end of the simulation, MOTION reads the final log file, parses it, and stores the collected results in a csv file. Web pages, with the complete package, can be found at https://sourceforge.net/projects/motion-project/.

4.2 Defining the mobility model

A realistic simulation of a MANET should take into account all its features. We have decided that the movement issues, as well as the amplitude of the radio range, are defined within the mobility model. We assume that the whole network topology is expressed by the connections among devices, implicitly, and

for each of them we consider only its current neighborhood. More precisely, in MOTION the network topology is expressed by an *adjacency matrix C*, such that $c_{ij} = 1$ if i and j are neighbors, 0 otherwise, for each pair of devices i and j. This implies that we can use concepts and properties of graph theory; 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.

Within MOTION, the mobility model is implemented into a Java class that, before executing any ASM move, updates the adjacency matrix. To this end, each c_{ij} is randomly set to 0 or 1, according to a mobility parameter defined by the user (see Section 4.4). The new values of the matrix are then set within the AsmetaL file, so that the ASM move can be executed, accordingly.

4.3 The Abstract State Machine-based models

The AODV routing protocol has been formally modelled through ASMs in [6] (Chapter 6). It is described as a set of nodes, each one representing a device. A modified version is used in MOTION, that takes into consideration the parameter *Timeout* (that is, the waiting time for the route-reply packet). The high-level definition of MOTION for AODV is:

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

where

```
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
        WaitingForRouteTo(a, dest) := false
  if WishToInitiate(a) then PREPARECOMM
  if not Empty(Message) then ROUTER
```

If the device needs to start a communication (i.e. the predicate *WishToInitiate* evaluates to true), then PREPARECOMM is called. The predicate *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. Finally, if the device has received a message (either RREQ, RREP or RERR), ROUTER is called, with

```
ROUTER = ProcessRouteReq
         ProcessRouteRep
         ProcessRouteErr
```

where each process expresses the behavior of the device, depending on the type of the message received.

The main difference between the previous model and the ASM model for N-AODV concerns ROUTER, that includes the call to PROCESS-NACK, in order to unicast the NACK packet, if needed.

The BN-AODV model is more structured, because it has to describe the behavior of three different kinds of agents: honest hosts, black holes, and colluders. So, the main rule has the form:

```
MAIN RULE BN-AODV=
    forall a ∈ Blackhole do BLACKHOLESPEC(a)
    forall a ∈ Colluder do COLLUDERSPEC(a)
    forall a ∈ Honest do HONESTSPEC(a)
```

where HONESTSPEC describes the behavior of the honest nodes, and it's analogous to AODVSPEC. BLACKHOLESPEC and COLLUDERSPEC are the specifications for the non-honest nodes and the colluders, respectively. Moreover, ROUTER for the honest nodes must verify the trustworthiness of the received RREPs.

Thanks to the formalization of the protocols, some correctness properties have been proved in the past, such as the starvation freeness for the AODV protocol, the properness of the packet (either NACK or RREP) received back by the initiator of any communication, when it is not isolated for N-AODV, and the capability to intercept blackhole attacks for BN-AODV.

4.4 Specific behavior of the tool

A 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 MANET includes a number of devices defined by the user, that depends on the specific evolution of the network (due to movements, some of them can be disconnected). Moreover, during each session, each device is the initiator for a number of attempts for establishing a communication, each of them towards a destination different from the initiator itself: the user expresses the probability that each device acts as an initiator 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 simultaneously executed. A communication attempt is considered successful if the initiator receives an RREP packet within the waiting time expressed by the parameter *RREP Timeout*; otherwise, the attempt is considered failed.

In MOTION, the devices 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 device is directly linked to any other. During the simulation, the devices mobility is expressed by the random redefinition of the values of the *adjacency matrix C*. More precisely, for each pair of devices $< a_i, a_j >$, and for each move of the ASM, the values of C are changed with a probability expressed by *Mobility level*.

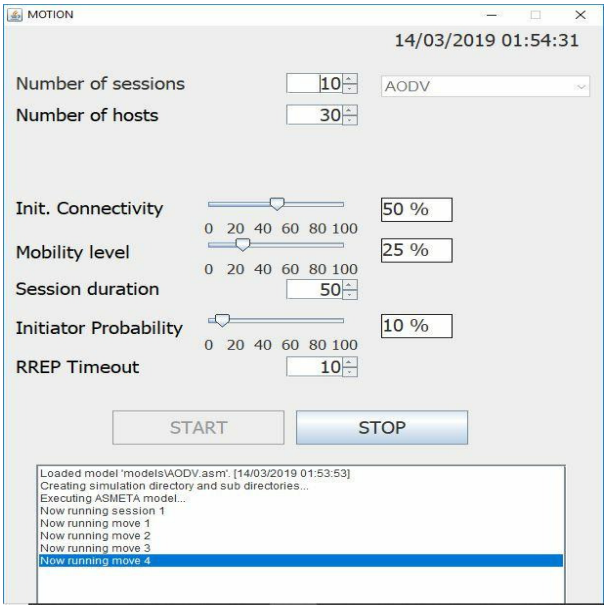

Fig. 1. MOTION user interface for AODV protocol

When the BN-AODV routing protocol is simulated, the MOTION user interface includes the definition of the number of black holes and colluders, and two parameters establishing the increment of the fake sequence number produced by the black holes. Figure 1 shows the current state of the simulation in the panel under the two buttons START and STOP.

From the ASM perspective, there are two different machines, both called by the ASMETA's main rule. The first one is OBSERVERPROGRAM: it is not part of the MANET, but it is used in order to manage the execution. It initializes the locations and data structures for all devices, manages the mobility (setting the initial topology and resetting the adjacency matrix at each move), and updates the counter for the time expiration. The second machine, called by the main rule, is the model of the devices' behavior. Currently, MOTION allows the users to study AODV, N-AODV, and BN-AODV; for all of them, the MANET is modeled by means of a Distributed ASM. In both AODV and N-AODV all the nodes behave in the same way, described by the respective DASM, so the machine specifying the protocol is called; at each move the machine randomly decides if the current agent will initiate new communication attempts by invoking PREPARECOMM, then it acts as a router by processing the proper control packets (with ROUTER).

5 Conclusions and future work

Mobile Ad-hoc NETwork 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 of these networks, 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 performance through direct simulation, but they aren't suitable to describe the properties of the networks. On the other hand, formal methods can do it, but they can hardly be used for studying performance properties.

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 the simulation engine AsmentaS. 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 Vector protocol and on two of its variants (concerning the host's network topology awareness and the ability to intercept blackhole attacks). Note that MOTION itself has been developed within the ASMETA framework, thanks to the abstract syntax defined in the AsmM metamodel.

A sensible improvement of MOTION could be the definition of a new interface, in which the dynamic evolution of the network, during the computations, is shown (as in [23]). Moreover, a complexity analysis of the network's protocols and the related algorithms could be performed, when the network is represented by means of ASM's. Finally, 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.

References

1. D. P. Agrawal, Q.-A. Zeng. *Introduction to wireless and mobile systems*. Cengage learning - Fourth Edition, Boston, 2016.
2. P. Arcaini, A. Gargantini, E. Riccobene, 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, 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. A. Bianchi, S. Pizzutilo, G. Vessio. Preliminary description of nack-based ad-hoc on-demand distance vector routing protocol for MANETS. In 2014 *9th International Conference on Software Engineering and Applications* (ICSOFT-EA), pp. 500–505. IEEE, 2014.

5. A. Bianchi, S. Pizzutilo, G. Vessio. Intercepting blackhole attacks in manets: An ASM-based model. In *International Conference on Software Engineering and Formal Methods*, pp. 125–137. Springer, 2017.
6. E. Börger, 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
7. E. Börger, R. Stärk. *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer Verlag, Berlin, 2003.
8. L. Bononi, G. D'Angelo, L. Donatiello. Hla-based adaptive distributed simulation of wireless mobile systems. In *Proceedings of the seventeenth workshop on Parallel and distributed simulation*, p. 40, IEEE Computer Society, 2003.
9. R. Calinescu, C. Ghezzi, M. Kwiatkowska, R. Mirandola. Self-adaptive software needs quantitative verification at runtime. *Communications of the ACM*, vol. 55(9), pp. 69–77, 2012.
10. N. Das, S. K. Bisoy, S. Tanty. Performance analysis of TCP variants using routing protocols of manet in grid topology. In *Cognitive Informatics and Soft Computing*, pp. 239–245, Springer, 2019.
11. F. Erbas, K. Kyamakya, K. Jobmann. Modelling and performance analysis of a novel position-based reliable unicast and multicast routing method using coloured Petri nets. In 2003 *IEEE 58th Vehicular Technology Conference*. VTC 2003-Fall, Vol. 5, pp. 3099–3104, IEEE, 2003.
12. A. Fehnker, R. van Glabbeek, P. Höfner, A. McIver, M. Portmann, W. L. Tan. A process algebra for wireless mesh networks In *European Symposium on Programming*, pp. 295–315, Springer, 2012.
13. A. Garcia-Santiago, J. Castaneda-Camacho, J. F. Guerrero-Castellanos, G. Mino-Aguilar, 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.
14. A. Gargantini, E. Riccobene, 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
15. U. Glässer, Y. Gurevich, 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
16. K. Jensen, L. M. Kristensen, 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.
17. N. Kaur, R. Singhai. Analysis of traffc impact on proposed congestion control scheme in AODV. *Wireless Personal Communications*, pp. 1–24, 2019.
18. C. Kim, E. Talipov, B. Ahn. A reverse AODV routing protocol in ad hoc mobile networks. In *International Conference on Embedded and Ubiquitous Computing*, pp. 522–531, Springer, 2006.
19. X. Li, M. R. Lyu, 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.
20. M. Merro. An observational theory for mobile ad hoc networks. *Information and Computation*, vol. 207(2), pp. 194–208, 2009.
21. A. P. Pandian, J. I.-Z. Chen, Z. A. Baig Sustainable mobile networks and its applications. *Mobile networks and application*, vol. 24(2), pp. 295–297, 2019.

22. C. E. Perkins, E. M. Belding-Royer, 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.
23. N. Saquib, Md. Sabbir Rahman Sakib, Al-Sakib Khan Pathan. ViSim: A user-friendly graphical simulation tool for performance analysis of MANET routing protocols. *Mathematical and Computer Modelling*, Vol 53, 11–12, June 2011, pp. 2204-2218
24. A. Singh, C. Ramakrishnan, S. A. Smolka. A process calculus for mobile ad-hoc networks. *Science of Computer Programming* vol.75(6), pp. 440–469, 2010.
25. D. A. Tran, 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.2006.151, URL https://doi.org/10.1109/TPDS.2006.151.
26. F.-H. Tseng, L.-D. Chou, H.-C. Chao. A survey of black hole attacks in wireless mobile ad hoc networks. *Human-centric computing and information sciences*, 1,4, 2011.
27. J. Wu, 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.
28. C. Xiong, T. Murata, 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.
29. M. G. Zapata. Secure ad hoc on-demand distance vector routing. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(3), pp. 106–107, 2002.