Discrete Event Modeling Study of AODV Routing Protocol with DEVS-Suite for Scalability Evaluation

Sinan Tuncel
Department of Computer Science Education
Sakarya University
54187 Sakarya - TURKEY
stuncel@sakarya.edu.tr

Ahmet Zengin
Department of Computer Science Education
Sakarya University
54187 Sakarya - TURKEY
azengin@sakarya.edu.tr

Hüseyin Ekiz
Department of Computer Science Education
Sakarya University
54187 Sakarya - TURKEY
ekiz@sakarya.edu.tr

ABSTRACT

This paper presents a robust simulation environment targeted for researching the complex dynamics of wireless computer networks. The general-purpose `DEVS-Suite` simulator supports animation with I/O and state trajectories of wireless computer network models developed using parallel DEVS modeling approach. The simulator offers high-level model abstraction as compared with simulators such as `ns-2`, Omnet++ and OPNET. The combined capabilities afforded by the robust `DEVS-Suite` simulator assists in understanding the fundamentals of wireless network topologies and the logics of wireless communication protocols. Large-scale wireless network models can be simulated and evaluated to show the benefits of DEVS formalism performance.

Categories and Subject Descriptors

H.4 [**Component-based simulation**]: Education and trainingNetwork modelsSimulation tools; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

—

Keywords

AODV, Modeling and Simulation, DEVS, DEVS - Suite, Scalability

1. INTRODUCTION

The Ad-Hoc On-demand Distance Vector (AODV) routing protocol is a widely used routing protocol for mobile ad-hoc networking [10]. In order to test and evaluate an ad hoc routing protocol in a real world environment and mostly in a large scale simulation scenario, utilization of modeling and simulation tools is inevitable. Today's most famous simulators from commercial vendors such as OPNET[9] and from academic area as such ns-2[8] and OMNET++[17] are commonly used by network research community. But, commerciality, lacks of system theoretic background, need for modular and hierarchical design, parallelism and scalability issues lead network researchers to develop new modeling strategies and high performance simulators.

In order to meet current network systems' demands, a new discrete event model of AODV routing protocol is developed based on DEVS formalism. DEVS formalism renders possible to establish proper and high performance simulation systems[19]. In this study, an AODV routing protocol for wireless ad hoc network systems is implemented based on DEVS-Suite, especially for large scale simulation scenarios and main features of developed simulation summarized.

`DEVS-Suite` is a general-purpose, discrete event simulation environment which supports visualization and tracking capabilities [6]. This is the new generation of the DEVSJAVA simulator [1] based on DEVS formalism [19]. This simulator also supports variable structure modeling [5]. The `DEVS-Suite` user-interface provides a consistent, efficient, integrated hierarchical component-based representation of models with run-time I/O and state trajectories and tabular data visualization. The AODV models developed on top of `DEVS-Suite` is the result of using networking theory as well as software engineering principles. Particular attention is paid to reliability and maintainability in view of the `ns-2` simulator. With the developed AODV simulator, users can create arbitrary network topologies, experiment with the models, and in particular track the dynamics of the network related to routing. `DEVS-Suite` simulator can be run on a personal computer as well as online via `DEVS-Suite Web Start` [12] which enables e-learning using Java Web Start technology.

In the remainder of this paper, starting in Section 2, presents the State of Art and description of the modeling concepts of DEVS and DEVS-Suite network simulator. In Section 3, the technics and ideas behind DEVS-Suite AODV simulator is given together with some features of it. In Section 4, ongoing research is summarized and we summarize our work and present some future research directions in Section 5.

2. BACKGROUND

2.1 DEVS Formalism

Network systems exhibit very high level complex, dynamic and parallel characteristics. Due to this fact, its complex yet

distributed behavior makes modeling effort of the networks difficult. However, discrete event modeling bringing abstraction and simplification mechanisms to modeling and simulation discipline facilitates modeling and simulation study systems such as computer networks demonstrating complex, dynamic, distributed and unpredicted behavior. The dynamics of network systems can be described using discrete event modeling. This is because the dynamics of the network systems can be characterized in terms of components that can process and generate events. Among discrete event modeling approaches, the Discrete Event Systems Specification (DEVS) [19] is well suited for formally describing concurrent processing and the event-driven nature of arbitrary configuration of nodes and links forming network systems. This modeling approach supports hierarchical modular model construction, distributed execution, and therefore characterizing complex, large-scale systems with atomic and coupled models. Atomic models represent the structure and behavior of individual components via inputs (X), outputs (Y), states (S), and functions. An atomic model can be described with,

$$\text{Atomic model} = (X, S, Y, \delta ext, \delta int, \delta conf, \lambda, ta).$$

The external (δext), internal (δint), confluent ($\delta conf$), output (λ), and time advance functions (ta) define a component's behavior over time. Internal and external transition functions describe autonomous behavior and response to external stimuli, respectively. The time advance function represents the passage of time. Output function is used to generate output messages sent through the output ports of atomic models. Atomic models receive messages which may cause a series of state transitions and output messages generated for consumption by other atomic or coupled models.

Atomic models can be coupled together in a strict hierarchy to form more complex models. Parallel DEVS, which extends the classical DEVS, is capable of processing multiple input events and concurrent occurrences of internal and external transition functions. The Parallel DEVS confluent transition function provides local control by handling simultaneous internal and external transition functions. A coupled model can be constructed by composing models into hierarchical tree structures. A coupled model is defined in terms of its constituent atomic and/or coupled models.

A coupled model can be constructed by composing models into hierarchical tree structures, and is defined in terms of its constituent (atomic and/or coupled) models. Connections between different atomic models can be performed by a coupled model (CM)[3], [19],

$$\text{Coupled model (CM)} = \\ < X, Y, D, M_d | d \epsilon D, EIC, EOC, IC >$$

The input and output sets X and Y have the same specification as those of the atomic model. D is a set of component names and M_d is set of atomic and/or coupled components, and EIC, EOC, and IC are external input, external output, and internal couplings, respectively. The closure under coupling feature allows a coupled model to be used as an atomic model when constructing other coupled models. Coupled models can be constructed systematically using the concepts of ports and couplings between them. When a component sends messages, the (external input, external output, and internal) couplings between input and output ports immediately relay the messages from the sender to receiver components. Upon receipt of messages by atomic models, the messages are processed, which may result in new states and generation of new outputs for other models. Parallel DEVS is capable of processing multiple input events and provides control for handling simultaneous internal and external events. Mathematical DEVS atomic and coupled models can be concretized in terms of UML. There exist other implementations of the DEVS specification approach based on single and multiprocessor environments. Parallel and distributed environments have been developed using technologies such as HLA [7].

The formal foundation of DEVS, its efficient execution, and the availability of sequential, parallel, or distributed simulation engines using alternative computational environments such as CORBA, HLA, and Web-services are important considerations. Furthermore, the DEVS models are extended with other kinds of models such as fuzzy logic [13].

2.2 DEVS-Suite

DEVS-Suite [6] is the discrete event general purpose simulation environment based on DEVS formalism and also is new version of the DEVSJAVA simulator[1]. In addition of single visualization function of DEVSJAVA, it adds additional functionality in the form of more tracking capabilities. DEVS-Suite has some constituent modules such as Simview, Timeview, DEVS Tracking Environment [15]. DEVS-Suite can simulate to models specified using the DEVS formalism [19]. The center piece of the simulator environment is Model Facade View Control (MFVC) by which simulation data can be displayed with its animation and viewing of time trajectories separated from the parallel DEVS abstract simulator.

In DEVS-Suite, execution of the models can be tracked as both the animation of the input/output messages for coupled models and the state changes of the atomic models, as well as log files. Simulation experiments can be triggered with test input which are can be selected via a dialogue box at the beginning of the simulation and time-based trajectories generated during simulation. At the end of the simulation, statistical outputs and trajectories can also be obtained for pre-defined phase and sigma state variables. Simulator has also an option window when loading the model which includes simview and tracking options. As already mentioned, simview is inherited from DEVSJAVA that provides visualization of DEVS models. However, it is clear that visualization usually decrease of the performance of a software system due fact of high resource demands of visual components and animation schemes. In DEVS-Suite simulation modeler, one can toggle the visualization or tracking capability of the package in case of need for high performance in large-scale experiments.

2.3 AODV Summary

AODV routing protocol provides unicast, broadcast, and multicast communication in ad hoc mobile networks [10].

AODV starts a route discovery when a route is needed by a source node or a node needs to join a multicast node group. Much of the complexity of the AODV protocol is to decrease the number of messages to conserve the capacity of the network. The routes are always loop-free through the use of sequence numbers. Nodes use this sequence number so that they do not repeat route requests that they have already passed on. Another such feature is that the route requests have a "time to live" number that limits how many times they can be retransmitted. If a route request fails, another route request may not be sent until twice as much time has passed as the timeout of the previous route request. The nodes running AODV maintain a routing table in which next hop routing information for destination nodes is stored.

In order to model AODV routing protocol and wireless networks, many researches have done including DEVS-based approaches such as [14], [4] and [11]. In addition to DEVS based approaches, some modeling effort of AODV protocol has been done as such [18] for power aware wireless networks,[16] using $\omega-$calculus approach and [2] using timed automata.

3. DEVS-SUITE-AODV FRAMEWORK

In order to model a wireless network system accommodating routing protocols, architecture can be split into three categories: topology, communication and the mobility. Topology has done with DEVS coupled models including nodes and radio channels, communication is held by network packets stretched from DEVS entities and finally mobility is overcome by initial and final coordinates (m,m) and speed (m/s) fields of every atomic model.

AODV simulation framework is based on the DEVS-Suite simulator and written in Java programming language (see Figure 1). Discrete event simulation system process the consecutive events to change the system's state. In our implementation, events are defined in terms of packet transmission, mobility and topological errors (see node atomic model definition in Appendix A). Events in developed system are processed by DEVS Engine. DEVS Engine serves as an operating system for events in system under observation. DEVS can process the events in a parallel manner.

Due to framework is developed purely in Java, all configurations, environmental and setup parameters and network layout are specified in Java code files, but configuration files can be used. In the following sections, we summarize the basic components of developed system.

3.1 Nodes and radio channels model descriptions

In a wireless network, nodes and communication channels are modeled basic processing units (i.e. routing intelligence in the network is done by these components). These models in the form of network coupled model conform to the Parallel DEVS atomic and coupled model specification and are implemented in the DEVS-Suite environment as seen in Figure 3.

Generic node model implements AODV routing protocol behavior defined using set theory as listed in Appendix A.

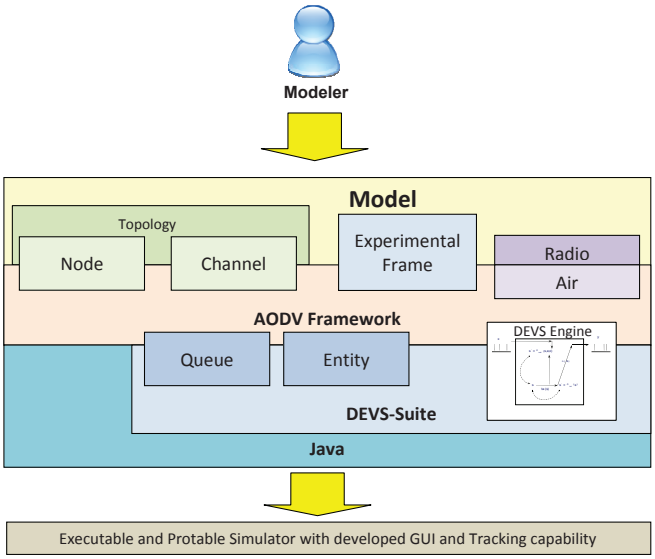

Figure 1: DEVS-Suite AODV System Structure

Wireless node model is typically composed of several modules as depicted in Figure 2. These modules are routing, media access and topography modules. Routing modules provide basic functionality for implementation of routing protocols such as AODV. Its architecture is ultimately generic so that every kind of routing protocols can be readily implemented. Module has different data structures such as routing tables and buffers to keep track of routing packets historically (see Figure 2). Also, each node stores own sequential number and the number of RREQ attempts.

One of the main features of DEVS formalism is to establish modular and hierarchical construction[19]. In this study, this feature is highly utilized in designing network components. Components design and selection is based on design and simulation objectives. Due for scalability, we selected high level abstractions and assumptions on designing the components. Lower memory and CPU utilize is preferable in large-scale experiments. Rather routing module, media access module is designed making several assumptions such as simplification of MAC protocol, implementing basic FIFO queue and 2D topography. In the media access component, physical channel model simulates the connection of devices, data storage in the buffers and link delays.

Location of a node that effects neighborhood and communication is determined by x and y values in the fixed size topography. Speed value defines node's mobility variance and every node is instantiated by start point (x_0, y_0) and final point(x_n, y_n) where movement is completed.

3.2 Modeling AODV with DEVS

Ad Hoc On-Demand Distance Vector (AODV) routing protocol belonging to distance vector routing algorithms family is a reactive behaviored protocol in which routes is calculated just only when a new data packet to be sent[]. In other words, it discovers the routes in the wireless network only when required. In this system, every nodes maintain

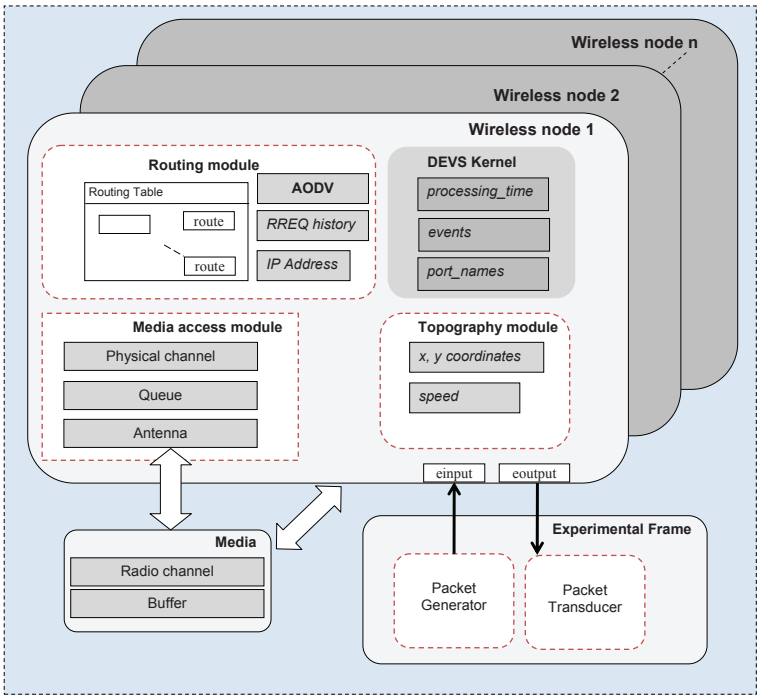

Figure 2: A DEVS-based wireless node is connected to other nodes via medium and emergent network is stimulated by an experimental frame to crate applications.

own routing tables in which discovered routes is kept with time stamp or version. In the background section, detailed information of AODV is presented.

Full behavior of the AODV protocol specified in [10] is applied using DEVS modeling and simulation formalism (see Appendix A). By extracting event based behavior of AODV from [10], some rules can be ordered and implemented in event based manner. Main event is sending and receiving packets in such a distributed system. However, channel down, battery exhaust, mobility change, and node congestions should be taken into considerations to mimic full behavior of a wireless system. As being in inner structure of a node, event abstraction is needed in modeling a highly complex, distributed systems. In this work, some abstractions are made including physical layer and antennas simplifications and two dimensional topography. These all abstractions may not actually reflect any real network operation, but highly needed yet because it is impossible to build exact virtual worlds in today's limited computing environments without making assumptions.

A network running AODV is modeled as interconnecting nodes in parallel composition of DEVS atomic models in DEVS-Suite. The states and interfaces of the nodes are initialized so that they mimic conditions of the real nodes placed at the beginning, after than nodes can get information from neighbors and begin to learn about the network. The most important characteristic of the AODV is route discovery in the presence of mobility on taken a data packet.

In AODV protocol life, a network node that needs a connection to specific destination creates a request for connection as RREQ messages. After receiving RREQ, remaining AODV nodes forward RREQ message, and update their routing table entry of that node that they receive it from. When a node receives such a message and already has a route to the desired node, it sends a message back to originating nodes named RREP message. The requesting node then begins transmitting the data using discovered route that it is shortest path and has least number of hops through other nodes. When a link fails, a routing error message RRER is passed back to a transmitting node, and the process repeats.

The main benefit of AODV routing protocol is that it causes no traffic for communication over links. Communication is set up on demand when two or more nodes need to data exchange. Besides distance vector routing approach is light weight algorithm and needs less computation resources. But AODV causes more time delay in establishing a connection, and this is done heavier that other protocols.

3.3 AODV messages

To implement the behavior of AODV protocol, some messages are modeled as DEVS entities such as Data, Hello, Route Request (RREQ), Route Reply (RREP), Route Error (RRER) and acknowledgement. These messages are exchanging between nodes and derived from entity class in DEVS models library. Data can embedded as an object to Data packets to be routed and remaining packets are all control packets.

3.4 Visualization

The four complementary views – component and message animation, time-based trajectories, SensorView animating wireless systems dynamism and tree listing of the model –

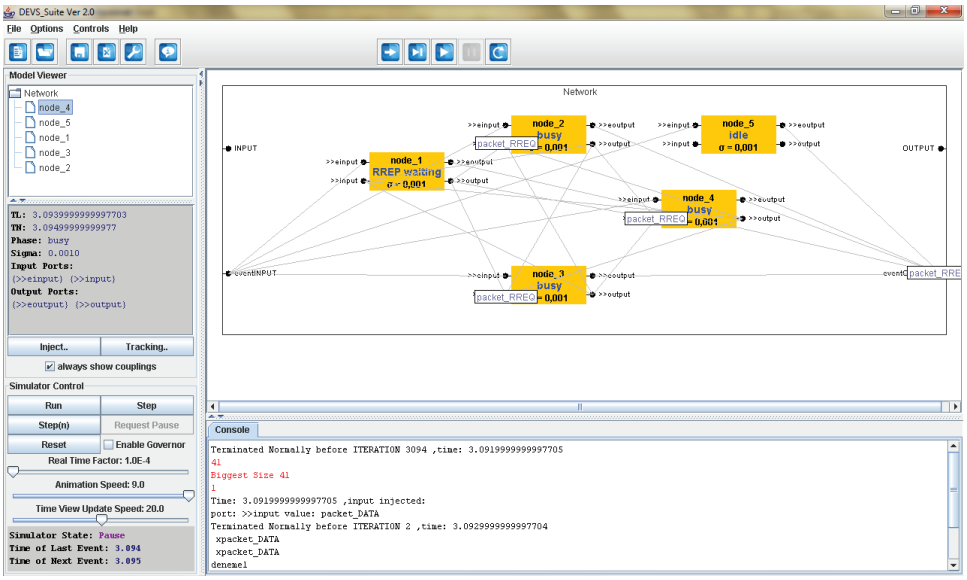

Figure 3: AODV network on DEVS-Suite Simview

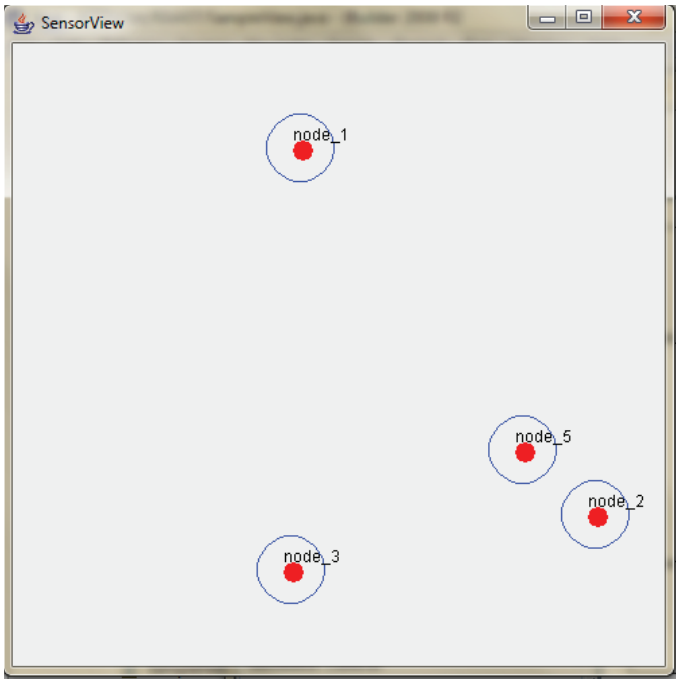

Figure 4: SensorView visualizing nodes on topography and animates mobility

provide a rich basis for researchers as well as students and teachers to view the structural and behavioral aspects of models. The routing protocol can be analyzed step-by-step through animation of nodes and transmission of messages. User selected inputs, outputs, and pre-defined state variables can be plotted as time trajectories. In Figure 3, five wireless nodes communicating via channels are shown. Models and their states, input and output ports (Network Interface Cards – NIC), and couplings, as well as traveling packets can be seen. Also, users can examine the composition hierarchy structure of models. Using `DEVS-Suite`, its possible to track and view the logic behind the routing protocol as well as the discrete event-based run of the wireless network system. Capabilities of the developed simulator can be extend to cover the protocols of the other layers in the OSI network reference model.

SensorView is a packet flow and network that developed as an extension to DEVS-Suite (see Figure 4). SensorView can show a network topology, nodes locations, links between nodes, demonstrate the flow of messages send upon links in a topography.

4. WORK IN PROGRESS

We plan to work to continue refinement process of the AODV model. Developed model needs to be verified and validated. In that phase we will use ns-2 network simulator for comparison of results from same network configurations. We extend to small scale models from a few nodes to thousands nodes large-scale models. The approach to store packets during route discovery needs further enhancements. Its possible to store them into the experimental framework. Finally we plan to develop on the shelf tools to create both large models and huge traffic to be run on the models.

5. CONCLUSIONS

We have presented a DEVS model for AODV routing protocol and implementing it as a shell on top of DEVS-Suite

kernel. DEVS-Suite has emerged to be a good discrete event simulator enables modeler to build models in system theoretic manner and provides more tracking capability and visualization. The DEVS simulation framework has been proven to be sufficient enough to meet the wireless systems requirements. Since framework's components are developed generic, it can be used for modeling MANETs as well as wireless sensor networks. Developed models will be publicly available on Sourceforge DEVS-Suite project site. Our AODV implementation will be continued to be refined.

6. ACKNOWLEDGMENTS

This work has been funded by the Sakarya University `Scientific Research Projects Agency` under contract `2007-05-02-001`. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Sakarya University.

7. REFERENCES

[1] ACIMS. DEVSJAVA modeling and simulation tool. `http://www.acims.arizona.edu/SOFTWARE`, 2009.

[2] S. Chiyangwa and M. Kwiatkowska. Modelling ad hoc on-demand distance vector (AODV) protocol with timed automata. In M. Leuschel, S. Gruner, and S. L. Presti, editors, *Proc. 3rd Workshop on Automated Verification of Critical Systems (AVoCS'03)*, Technical Report DSSE-TR-2003-2, University of Southampton, April 2003.

[3] A. Chow. Parallel DEVS: A parallel, hierarchical, modular modeling formalism and its distributed simulator. *Transactions of the Society for Computer Simulation International*, 13(2):55–67, 1996.

[4] U. Farooq, G. Wainer, and B. Balya. Devs modeling of mobile wireless ad hoc networks. *Simulation Modelling Practice and Theory*, 15(3):285 – 314, 2007.

[5] X. Hu, B. P. Zeigler, and S. Mittal. Variable structure in DEVS component-based modeling and simulation. *SIMULATION: Transactions of The Society for Modeling and Simulation International*, 81(2):91–102, 2005.

[6] S. Kim, H. Sarjoughian, and V. Elamvazhuthi. Devs-suite: A simulator supporting visual experimentation design and behavior monitoring. In *Proceedings of the Spring Simulation Conference*, pages 29–36, San Diego, CA, March 2009.

[7] Y. J. Kim, J. H. Kim, and T. G. Kim. Heterogeneous simulation framework using DEVS BUS. *Simulation*, 79:3–18, 2003.

[8] NS-2. Network simulator. http://www.isis.edu/nsnam/ns/, 2009.

[9] OPNET. Opnet network simulator. http://www.opnet.com, 2009.

[10] C. Perkins, E. Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. Internet Engineering Task Force, http://www.ietf.org/rfc/rfc3561.txt, 2009.

[11] A. T. Santoni, J. F. Santucci, E. De Gentili, and B. Costa. Discrete event modeling and simulation of wireless sensor network performance. *Simulation*, 84(2-3):103–121, 2008.

[12] H. Sarjoughian. Devs-suite webstart. http://acims1.eas.asu.edu/WebStarts/, 2009.

[13] H. Sarjoughian and F. Cellier. *Discrete Event Modeling & Simulation Technologies: A Tapestry of Systems and AI-based Theories and Methodologies for Modeling and Simulation*. Springer Verlag, 2001.

[14] H. Sarjoughian and K. Shaukat. A comparative study of DEVS and ns-2 modeling approaches. *Transactions of the Society for Modeling and Simulation International*, 2009.

[15] H. Sarjoughian and R. Singh. Building simulation modeling environments using systems theory and software architecture principles. In *Proceedings of the Advanced Simulation Technology Conference*, pages 99–104, Washington DC, April 2004.

[16] A. Singh, C. Ramakrishnan, and S. Smolka. Modelng the aodv routing protocol in the Â£-calculus. In *Systems, Applications and Technology Conference, 2006. LISAT 2006. IEEE Long Island*, pages 1–5, May 2006.

[17] A. Varga. The omnet++ discrete event simulation system. http://www.omnetpp.org/, 2009.

[18] D. Weber, J. Glaser, and S. Mahlknecht. Discrete event simulation framework for power aware wireless sensor networks. In *Industrial Informatics, 2007 5th IEEE International Conference on*, volume 1, pages 335–340, June 2007.

[19] B. P. Zeigler, H. Praehofer, and T. G. Kim. *Theory of Modeling and Simulation*. Academic Press, New York, 2000.

APPENDIX

A. DEVS SPECIFICATION OF AODV PROTOCOL

In this section, specifications for the atomic wireless node are given below. These models conform to the Parallel DEVS specification and are implemented in the DEVS-Suite environment.

M_{sensor_node} = <X , Y , S, δext, δint, δcon, λ , ta>

where

Input ports and values

X = inport × invalues

inports : {input, einput}

invalues : {packet, packet_DATA, packet_HELLO, packet_RREQ, packet_RREP, packet_RERR, packet_RREP_ACK, packet_hold},

Output ports and values

Y = outport × outvalues

outports : {output, eoutput},

outvalues : {packet, packet_DATA, packet_HELLO, packet_RREQ, packet_RREP, packet_RERR, packet_RREP_ACK, packet_hold},

State sets

S= phase $\times\sigma \times \bar{Q}$

phase : {setup, busy, idle, congested, discarded, data_received,
RREP_waiting, RREQ_receiving, RREP_generated, RREP_update&forward,
RREP_received, neighbor_added}

$\sigma = \Re^{+}_{0,\infty}$,

$Q = Q_{queue}$

where queue is for incoming and outgoing packets.

External transition function

$$\delta ext((phase, \sigma, \bar{Q}), e, x) =$$

$$\begin{cases} s \leftarrow discarded, \sigma, \bar{Q} & \text{if x = RREQ} \wedge \text{x} \in \text{RREQhistory} \\ s \leftarrow RREQ\_receiving, \sigma, \bar{Q} & \text{if x = RREQ and x} \ni \text{RREQhistory} \\ s \leftarrow RREP\_received, \sigma, \bar{Q} & \text{if x = RREP} \wedge \text{x'nextDestHop=adress} \wedge \text{x'destination=address} \\ s \leftarrow RREP\_update\&forward, \sigma, \bar{Q} & \text{if x = RREP} \wedge \text{x'nextDestHop=adress} \wedge \text{x'destination} \neq \text{address} \\ s \leftarrow discarded, \sigma, \bar{Q} & \text{if x = RREP} \wedge \text{x'nextDestHop}\neq\text{adress} \\ s \leftarrow busy, \sigma, \bar{Q} & \text{if x is enqueued to Q} \\ s \leftarrow congested, \sigma, \bar{Q} & \text{if x is not enqueued to Q} \\ where \quad s \in S \end{cases}$$

Internal transition function

$$\delta int(phase, \sigma, \bar{Q}) =$$

$$\begin{cases} \text{dequeue packet from } \bar{Q} \text{ and } s \leftarrow data\_received, \sigma, \bar{Q} & \text{if packet = DATA} \\ \text{add new route and } s \leftarrow neighbor\_added, \sigma, \bar{Q} & \text{if packet = Hello} \\ \text{add new route} & \text{if packet = RREQ} \wedge \text{packet's source is unknown} \\ s \leftarrow RREQ\_forward, \sigma, \bar{Q} & \text{if packet = RREQ} \wedge \text{packet's destination is unknown} \\ \text{update route and } s \leftarrow RREQ\_receiving, \sigma, \bar{Q} & \text{if packet = RREQ} \wedge \text{packet's seq. number} \geq \text{RT's seq. number} \\ \text{update route and } s \leftarrow RREP\_generate, \sigma, \bar{Q} & \text{if packet = RREQ} \wedge \text{packet's seq. number} \leq \text{RT's seq. number} \\ \text{add new route} & \text{if packet = RREP} \wedge \text{packet's source is unknown} \\ \text{update route and } s \leftarrow RREP\_update\$forward, \sigma, \bar{Q} & \text{if packet = RREP} \wedge \text{packet's source is known} \\ s \leftarrow RREP\_received, \sigma, \bar{Q} & \text{if packet=RREP} \wedge \text{packet's dest=address} \\ s \leftarrow idle, \sigma, \bar{Q} & \text{if Q is empty} \\ where \quad s \in S \end{cases}$$

Confluent transition function

$\delta con((phase, \sigma, \bar{Q}), e, x) = \delta ext(\delta int(phase, \sigma, \bar{Q}), 0, x)$

Output function

$$\lambda(phase, \sigma, \bar{Q}) = \begin{cases} y \leftarrow outports, Hello & \text{if phase = setup} \\ y \leftarrow outports, RREQ & \text{if phase = data\_received} \\ y \leftarrow outports, RREQ & \text{if phase = RREQ\_receiving} \\ y \leftarrow outports, RREP & \text{if phase = RREP\_generated} \\ y \leftarrow outports, RREP & \text{if phase = RREP\_update\&forward} \\ where \quad y \in Y \end{cases}$$

Time advance function

$ta(s) = \sigma$