Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

DEVS-SUITE SIMULATOR: A TOOL TEACHING NETWORK PROTOCOLS

Ahmet Zengin

Computer Science Department
Sakarya University Faculty of Technology
Sakarya, 54187, TURKEY

Hessam Sarjoughian

School of Computing Informatics
Decision Systems Engineering
Arizona State University, Tempe, AZ, USA

ABSTRACT

Understanding of the underlying concepts, principles, and theories of computer network can significantly benefit from simulation tools. Usage and development of simulation models for computer networks, however, can be demanding in educational settings. While a variety of open source and commercial tools are available, there remains a desire for simulators that can better support student learning and instructor teaching. In this work, the DEVS-Suite general-purpose simulator is extended to support modeling of network protocols. A model library for the OSPF protocol has been developed such that the emphasis is placed on education to capture the basic principles of network protocols using sound modeling and simulation principles instead of supporting highly detailed network protocol simulations. The use and pedagogical effectiveness of the DEVS-Suite network simulator is carried out in a classroom setting. The results of the student survey are presented and discussed.

1 INTRODUCTION

Computer networks including communication protocols (e.g., TCP/IP and OSPF) are essential for computers to communicate with each other. Although communication protocols can be simply described as a set of rules governing the syntax and semantics of computing resources, simulation modeling is necessary to understand their complex nodes. In particular, simulation tools are commonly used in teaching computer network principles and practices (e.g., (Guo, Xiang, and Wang 2007)). In response, a variety of tools (e.g., ns-2 (Simulator 2010), OMNeT++ (Varga 2010), OPNET (OPNET 2010)) have been developed to meet the needs of scientists and engineers primarily for research purposes. This is unlike simulation tools (e.g., Matlab®/Simulink®) that are commonly used for teaching engineering & science subjects.

A simulation tool may be classified as either domain-specific or domain-neutral. A domain-specific simulation tool has a library of domain-specific components that define models for a domain such as computer networks. Tools such as OPNET are targeted for domain experts and assume users, among other things, understand fundamentals of modeling and simulation concepts and methods. In contrast, a general-purpose simulator is developed based on a generic modeling and simulation theory or framework. An example of this kind of simulation tool is DEVS-Suite (DEVS 2010). To use such a tool for simulating computer networks, it is desirable to provide a library of model components such as processing nodes and communication protocols. Such a tool can be well suited for educational use since the rigorous DEVS model formalism play an important role in teaching and learning domain knowledge since the principles of modeling and simulation theory (e.g., separation of a model from its simulator) are already established (Chen and Sarjoughian 2010).

In this paper, we will introduce a domain-specific model component library for OSPF protocol into the DEVS-Suite simulator. In Section 2, selected network simulation tools and the OSPF protocol are briefly presented. In Section 3, DEVS-Suite simulator is described. In Section 4, network component models are developed in DEVS-Suite. In Section 5, an exemplar network protocol model is used to demonstrate the key features of the DEVS-Suite network simulator. In Section 6, the role of the simulator in student learning and teacher instruction in an educational setting is discussed. In Section 7, concluding remarks including future work is briefly presented.

2 BACKGROUND AND RELATED WORK

2.1 Network simulation tools

An important method for design and analysis of such systems and their routing protocols is simulation modeling, especially when analytical methods are known to be inapplicable (Floyd and Paxson 2001). Basically, network simulators try to model real world network systems and often are required since experiments may not be possible with actual computer networks. However, real network systems can be modeled by means of abstraction mechanism which renders possible to model complex systems into the limited resource computers. A network simulator should enable users to represent a network topology, prepare different scenarios, specify the nodes and links, and analyze the results. Graphical user interfaces allows the simulation users to visualize and track working of simulated environment.

The basic limitations of analytical approaches have led to a variety of modeling and simulation approaches and tools. Tools such as ns-2, ns-3 (ns 3 2010), OPNET, OMNeT++, Glomosim (Zeng, Bagrodia, and Gerla 1998) and SSFNet (Cowie, Ogielski, and Nicol 2002) are used to reveal the inner workings of computer networks in virtual settings. A key emphasis has been on enabling design and testing of routing algorithms, MAC layers, and end-to-end queuing. Although the capabilities of these simulation tools support describing (wired and wireless) computer and device network protocols and communications in great detail, their underlying foundation lacks support for developing models in system theoretic manner. The conceptual models of these tools are derived from computer network hardware and software abstractions. These models are mostly implemented in object-oriented programming languages and simulated in virtual and/or emulated in physical testbeds. But these tools have some disadvantages in terms of underlying methodology, implementation and scalability (Fujimoto et al. 2002). These reasons are limited or lack object-oriented concepts for designing and implementing simulators. They lack support for casting real world entities to non-object simulation components. Furthermore, scalable and efficient execution requires support for not only concurrent simulation execution methods but also simple, yet sound modeling concepts. That is despite tools such as pdns (Riley, Fujimoto, and Ammar 1999) and SSFNet being implemented in object oriented programming languages, they lack formal modeling theories. Furthermore, other tools such as OPNET are inherently not well suited for parallel and distributed execution. Other concerns about these simulators are limited or weak support for visualization as well as difficulties in tool installation and ease of use. Detailed comparisons of the network simulators are presented in (Begg et al. 2006) and (Lessmann et al. 2008).

2.2 OSPF Protocol

Routing protocols in the network systems can be split into two main categories: link state routing and distance vector routing. Currently, in particular for Internet, while distance vector protocols are used for inter-gateway interactions, link state protocols are used for intranet case (Steenstrup 1995). Open Shortest Path First (OSPF) as one of the famous link state routing protocol is an open standards routing protocol and a particularly efficient interior gateway (IGP) routing protocol that is faster than routing information protocol (RIP) which is one of the most known kinds of the distance vector protocols family. It employs the Dijkstra algorithm when estimating the shortest paths (Dijkstra 1959).

The OSPF routing protocol was developed to provide an alternative to RIP, based on Shortest Path First algorithms instead of the Bellman-Ford algorithm which is the basis for RIP. It uses a tree that describes the network topology to define the shortest path from each router to each destination address. Since OSPF routing protocol keeps track of entire paths, it has more overhead than RIP, but provides more options. The main difference between OSPF and RIP is that RIP only keeps track of the closest router for each destination address, while OSPF keeps track of a complete topological database of all connections in the local network. The OSPF algorithm works on three phases. They are important for describing how the algorithm behaves in discrete event fashion. In the *Startup* phase as soon as a router connects to the network, it sends Hello packets to all of its neighbors, receives their Hello packets in return, and establishes routing connections by synchronizing databases with neighbor routers that agree to synchronize. In the *Update* phase, each router sends an update message at regular intervals. This message is called its "link state" describing its routing database to all the other routers, so that all routers have the same description for the local network topology. Finally, in the *shortest path tree phase*, each router then estimates a mathematical data structure called a "shortest path tree" that describes the shortest path to every destination address indicating the closest router for communication.

3 DEVS-SUITE SIMULATOR

Network systems exhibit very high level complex, dynamic and parallel characteristics. Discrete event modeling brings abstraction and simplification mechanisms which facilitate modeling and simulation study of complex system dynamics. The dynamics of 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) (Zeigler et al. 2000, Chow 1996) is well suited for formally describing concurrent processing and the event-driven nature of arbitrary configuration of network systems. This modeling approach supports hierarchical modular model construction, distributed execution, and therefore characterizing complex, large-scale systems with atomic and coupled models.

DEVS-Suite (DEVS 2010, Kim, Sarjoughian, and Elamvazhuthi 2009) is an open source, general-purpose discrete event simulation environment supporting parallel DEVS models. It is the next generation of the DEVSJAVA simulator (ACIMS 2010) and supports execution of models using sequential and concurrent schemes. Hierarchical models can be displayed as components within component style with support for animation of I/O messages among model components and viewing of atomic model's states.

DEVS-Suite introduces new concepts and capabilities to support the complex task of simulation modeling. It offers on-the-fly configuration of experiments (GUI-based selection and monitoring of I/O and states) and visualization of each component's behavior as time-based trajectories as well as synchronization of animation and time trajectories (Helser 2009). This simulator consists of the DEVSJAVA simulator kernel and Simview for hierarchial viewing of models and their message-passing animation, Timeview for plotting time trajectories, DEVS Tracking Environment for automated design of experiments, and controller for time-stepped, combined visualization of component animation and time trajectories. The DEVS-Suite simulator core software architecture uses the Model Facade View Control (MFVC) which was first used in the DEVS Tracking Environment (Sarjoughian and Singh 2004). These capabilities along with support for use via web are important for education and distance learning (Sarjoughian 2010).

In DEVS-Suite, execution of the models can be tracked as both the animation of the input/output messages for atomic/coupled models and the state changes of the atomic models, as well as log files. Simulation experiments can be triggered with test input which 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 (Mather 2003) and tracking options. simview is inherited from DEVSJAVA that provides visualization of DEVS models. However, visualization can significantly decrease simulator's performance due to high demand for hardware resources. In DEVS-Suite, users can choose in any combination animation of model components, viewing of components' time trajectories as well as storing the observed date in a CSV file, or writing user-defined data in source code to console. The degree of flexility has important benefits including reducing execution time of large-scale experiments. Obviously, different users have different needs as they develop their simulation models and experiment with them. Hence, different needs of users can be satisfied given the simulator's different capabilities and flexibility.

4 DEVS MODEL FOR NETWORK PROTOCOL SIMULATION

In this section, we summarize some features of the DEVS-Suite network simulator. Detailed information of underlying modeling and simulation knowledge about developed system together with validation experiment can be found in (Zengin and Sarjoughian 2009). DEVS models of the other simulation protocols are implemented as such Routing Information Protocol (RIP) and Swarm Intelligence based protocols in (Zengin, Sarjoughian, and Ekiz 2004), AODV wireless protocol in (Farooq, Wainer, and Balya 2007) and wireless sensor networks in (Santoni et al. 2008). For handling model and simulation scalability, a high level model abstraction for real networks is adopted - e.g., MAC layer and acknowledgement packets details are carefully considered and simplified.

4.1 Basic Components

In order to develop a domain-specific modeling and simulation environment such as DEVS-Suite network simulator, it is important to define basic components such as nodes and links. The basic network components are depicted in Figure 1 in the form of interacting elements composing a network and its experimental frame. They represent application layers tasks. Since the main focus is aiding teaching and learning of network routing protocols, classical seven-layered approach is not considered. Instead, a flat hierarchical, but modular approach is chosen and the main components desired for protocol implementation such as queues, packets and routing tables are modeled. Atomic node behavior depends on interacting modules via messages that lead to whole design cross-layer interdependencies. Modular design allows integration of the new models easily and models designed based on DEVS atomic and coupled model specification are flexible to add enhancements. A coupled model can be used to create user defined network topologies in network system to evaluate network functionality and educational purposes. Coupled models is used to model Autonomous Systems (AS) in large networks.

To run the network under specific scenario, applications such as user traffic and errors have to be implemented. All transactions in terms of session and application layer's tasks are treated as events in network model. The experimental frame has simply an event generator and a transducer to generate different kinds of event and analyze

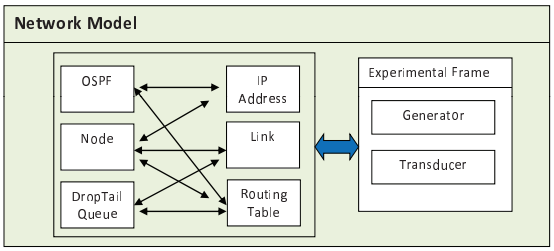


Figure 1: Network model components

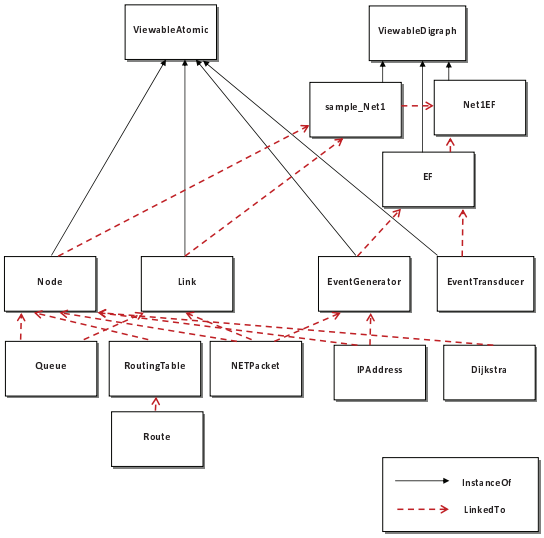


Figure 2: Network model class diagram

the outputs, respectively (see Figure 1). Transducer's observation capability together with the tracking environment provide enhanced control and observation over developed models under experimentation.

Figure 2 depicts a simplified class diagram for the DEVS-Suite network simulator. All processing models such as nodes, links, generator and transducer are derived from the viewable atomic classe. Atomic and digraph classes are derived from the "devs" class which supports generic structure and behavior (e.g., I/O interface, timing) for all atomic and coupled models developed for the DEVS-Suite network simulator. Dashed lines in the diagram show associations of the classes while solid lines depict dependency. Lowest level component is the Route class that defines the basis for evaluation and training of OSPF and other protocols.

4.1.1 OSPF dynamics

The DEVS-Suite OSPF implementation is developed on top of the DEVS-Suite kernel. DEVS-Suite OSPF environment exploits DEVS formalism and proven software engineering techniques to achieve performance, ease of deployment and use, accuracy, scalability and system theoretic design. As depicted in Figure 3, events processed by nodes and links can be described as statecharts. They are important for showing state changes in response to internal and external events and thus visualizing the behavior of simulation models. A node atomic model is born with "startup" phase and does not have any information for other model components. After sending a hello message to its neighbors, every node starts to create tables and learns what happens in the network. Ten events in the system are selected for educational and performance purposes. Increasing number of events decrease performance and increase accuracy. These ten events are sufficient for understanding the OSPF routing protocol logic. Only external and internal transition functions can cause a node to change its context for new events.

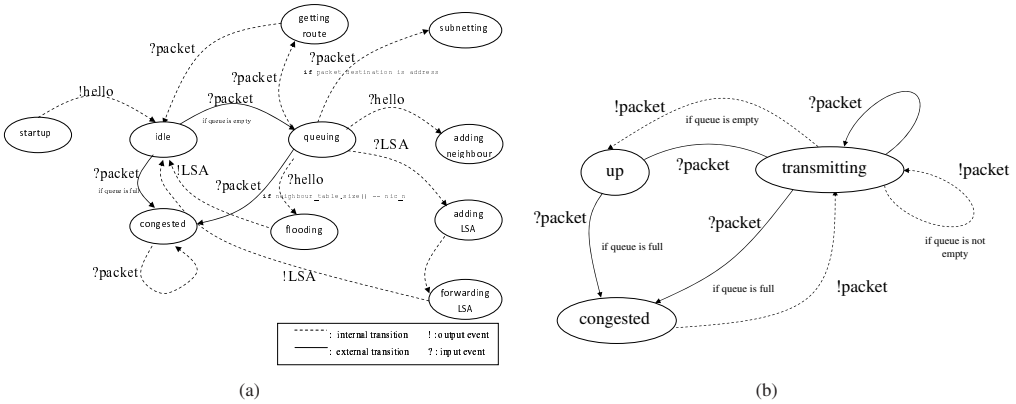


Figure 3: (a) Node states, state transitions and output, (b) Link states, state transitions and output.

4.2 Buffering

In developed modeling and simulation environment, queue is a Linked List data structure which maintains the FIFO (First In First Out) order of the objects inserted into it. Some new objects can be inserted and others removed, but the order of the objects in both states is maintained. Packets have different priorities which means insertion order of newly incoming packets is determined by priority value (0 lowest, 7 highest). The queue implements a very simple drop-tail queue with the following behavior.

- If packet's priority value has highest value, it is inserted at the front of the queue.
- If a packet is to be enqueued and is bigger than the maximum queue size, it is rejected.
- If a packet to be enqueued is smaller than the maximum queue size, but there is not enough space for it, the packets at the end of the queue are dropped until enough space becomes available.
- If there is enough space for a packet to be enqueued, it is inserted at the front of the queue.

5 OSPF PROTOCOL SIMULATION WITH DEVS-SUITE

In this section, we go step by step through how a model is loaded and simulated in DEVS-Suite network simulator. The emphasis is to describe how a model can be experimented with by focusing on key needs during the simulation execution, particularly in terms of visualization and tracking aspects.

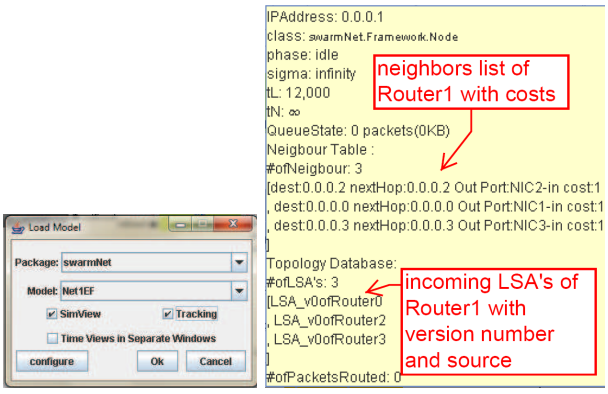
5.1 OSPF Protocol Visualization by Simview

5.1.1 Model loading options

The visualization capability is an optional property for simulation study in DEVS-Suite. It is important in particular when we simulate large-scale simulation systems due for visual components in a software system may occupy a great deal of computation resources. Hence, in general during research, a pure textual output of simulation results is preferred for the sake of performance. Visualization has an important role in education. A graphical representation of a simulated scenario can help to comprehend and track basic logic behind the routing protocol and whole network dynamics. Therefore, DEVS-Suite provides a toggle option while simulation model is loading (see Figure 4).

5.1.2 Model visualization

A modeler can view the entire implementation of a computer network as well as, for example, the OSPF protocol. Figure 5 depicts a network model consisting of four nodes, four links, one generator, and one transducer. A model component is represented by a rectangle, with input ports on the left and output ports on the right. Inside the box view of the model, current state, model name and its sigma value can be seen as binary strings. Couplings are represented by lines. The entities belonging to network systems such as routes, routing tables, packets and queues can also be visually tracked by developed simulator (see Figure 4). If the mouse is placed on a network node, a tooltiptext shows neighbors table and routing database of related node together with all user defined model states that are selected for viewing (see Figure 4). No such visualization is made available in ns-2 and OMNeT++.



(a) (b)

Figure 4: (a) DEVS-Suite model load pane toggling between simview and tracking visualization, (b) run-time viewing of routing table state.

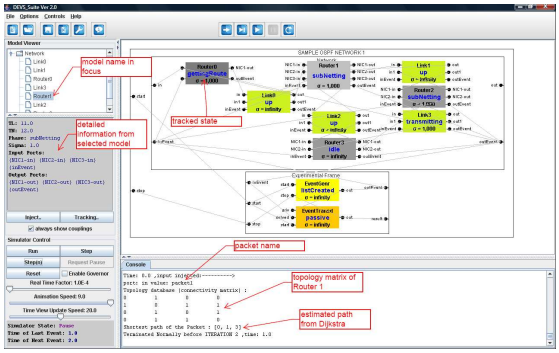


Figure 5: DEVS-Suite simulation viewer and a sample OSPF protocol simulation model.

5.1.3 Controls over simulation experiments

DEVS-Suite together with Tracking Environment provide complementary mechanisms for controlling simulation experiments. As seen in Figure 5, current simulation time is displayed at the bottom. A slider serves a real time factor at any time during the simulation. The simulation execution can be controlled using Run, Step, Step(n), Request Pause, and Reset (see Figure 5).

The other controls show coupling, real-time factor, animation speed, and TimeView update speed sliders. DEVS-Suite separates animation of messages and time-based trajectory plotting and thus provides flexibility in simulation speed vs. simulation visualization. Simulated model can be triggered manually with an input. Though generator atomic model in experimental frame can schedule events in automated manner, modeler can create an input event manually during simulation by clicking on the input ports, a small pane appears (DEVS 2010), than click step button to observe the change of the states and outputs. An input event consists of a port name, data value, and elapse time (e.g., port: in, value: packet1 packet, e: 0.0). The elapsed time is a time stamp of the associated event and is used to schedule and inject a specific event at some, finite, future time instance. It is in units of time after the current time, shown by simulator clock.

5.2 Envisage the discrete event logic of OSPF protocol in a virtual lab

It is important to track a network protocol's execution visually in teaching computer network concepts and theories (see Figure 6). Students have to master protocols' logic in a step by step manner. Though ns-2 provides a level of detail in nam visualization, it is impractical to understand much of the protocols details via animation. The DEVS-Suite network simulator allows students to comprehend the creation of routing databases as well as detailed behavior of the simulated communication protocol. The basic underlying details of the OSPF protocol simulation

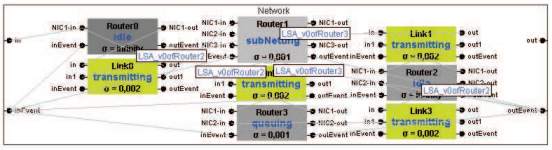


Figure 6: An animated view of the network exchanging the learned topology of LSA messages.

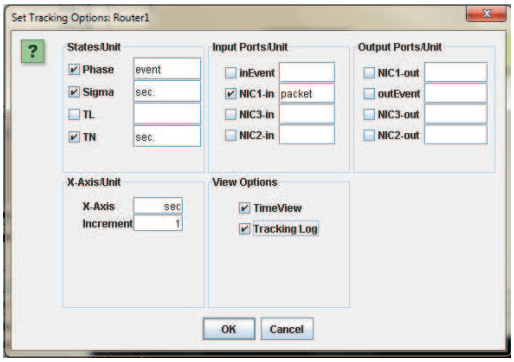


Figure 7: Tracking network model

model are described in (Zengin and Sarjoughian 2009). In the following, discretized execution of the OSPF protocol logic is divided into several stages and assisted with screenshots to demonstrate the educational role of the simulator.

As already mentioned in Section 2, a link-state protocol uses a neighbor table and a topology database in addition to adding routes to the routing table. In order to mimic the OSPF routing table creation, updating, and registration, the node model accommodates vector data objects for two data sets. Then these two objects pass to the Dijkstra class as parameters and therefore shortest path is generated (see Figure 5).

5.3 OSPF Protocol Tracking by Timeview

The TimeView module in DEVS-Suite is developed for runtime plotting of data sets as two dimensional plots and allows to display all dynamics and properties of every node, link atomic and network coupled model. As already mentioned, modelers have the flexibility to select animation and tracking view options for any number of atomic/coupled models. They can set the unit for data that is to be monitored as well as the time axis. The time increment, units, and the selection of data to be observed can be set as shown in Figure 7. Modeler is able to view inputs, outputs, states and time-based trajectories using TimeView. A few dynamics of the OSPF protocol model can be seen in Figure 8. These charts together with log files are useful in examining time-based operations of the OSPF protocol simulation components.

5.4 Manipulating Simulation Data

Data collection is one of the main process in any simulation study (Robertson and Perera 2002). Besides, data is used for measuring performance of the simulator, it can be employed for verification and validation processes. In addition, data collection is difficult particularly as the scale of the model increases. In that case, automated data collection become crucial. DEVS-Suite supports selection, manipulation and alternative viewing of simulation data sets.

As mentioned above, modeler can observe simulation data for any number of atomic and coupled models without writing any code. The data can also be obtained in the form of the tabular using a tracking logger. An alternative data presentation is to export these values to CSV files. After collecting data from the simulation, the analysis of network performance is done and commonly available tools can be used to customize and plot simulation results.

6 SIMULATORS' EVALUATION

To evaluate the effectiveness of the DEVS-Suite network simulator, it is used in a graduate course called Design and Simulation of Computer Networks. This course with 30 students is taught in the fall 2009 EBE507 at Sakarya

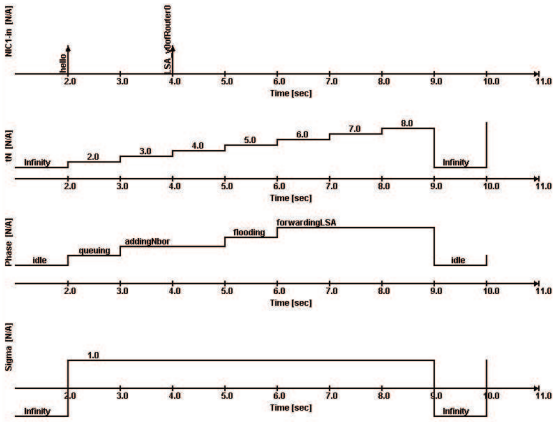


Figure 8: Trajectories of OSPF protocol model

University Technical Education Faculty in Turkey. The students are required to use the DEVS-Suite and ns-2 simulators in three of their homework assignments. The first assignment was to evaluate the deployment of the tools. During the first week of the semester, all students installed ns-2 on Cygwin (Windows machine). They copied the DEVS-Suite java archive (jar) on their flash memory devices. Students were required to evaluate and compare the deployment of the simulators. For the second assignment, students were asked to develop a network topology composed of 10 routers and 11 links both in the DEVS-Suite and ns-2 simulators with almost the same parameters and underlying protocols. Students were asked to observe the simulation runs and analyze their results. In this case, visualization and tracking capabilities of the simulators were the focus of the evaluation. For the third assignment, students studied scalability aspect of computer networks. Students needed to develop a network topology with a few thousands of nodes using a recursive algorithm and later coded with pure Java and oTcl. The students simulated these models under varying conditions (experimental settings), analyzed the results, and evaluated the strength and weaknesses of the DEVS-Suite and ns-2 simulators.

They were asked the following twelve questions. The survey questions are divided to Parts I, II and III. In Part I, the generic capabilities of the DEVS-Suite simulator were evaluated such as visualization and animation. In Part II, the degree of the simulator's support for teaching routing protocol, network structure and behavior were evaluated.

1. Did you find the simulator interface, layout, and collapsible panels simple and sufficient for your need?
2. Did you find the combination of component animation, time-based I/O and state trajectories views and console outputs useful and complementary?
3. Did you find the simulator's execution and animation speeds acceptable?
4. Did you find the simulator's control panel and icons (buttons) simple and easy to use?
5. Did you find the simulator's TimeView Update Speed useful?
6. Did you find the Enable Governor support useful?
7. Did the simulation of the OSPF increase your understanding of the theoretical principles of OSPF control packets flow, package queuing and delays?
8. Did you find configuration of experimental data simple to view?
9. Did you find creation of network composed of nodes and links together with their parameters easy?
10. Did you find the effects of link and node errors such as link down and node congestions tractable and understandable during simulation?
11. Did you find observing instantiations and updating of routing tables helpful in the understanding of the OSPF logic?
12. Did you find the assignment instructions explaining how to load and use the simulator clear and understandable?

In Part III, students compared the DEVS-Suite and ns-2 simulators (see Table 2). The questions focused on relative ease of use as a measure of learning each tool and using it effectively and quickly. Other important factors in evaluation of each tool were existence of model libraries, support for visualization, execution efficiency, and availability of documentation (most notably textbooks, technical references, user guides, and papers) detailing the fundamentals of modeling and simulation concepts and methods.

Table 1: Survey results of DEVS-Suite simulator and OSPF network protocol

Likert Scale	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Strongly Agree	27	27	62	34	15	-	46	30	30	15	42	23
Agree	65	54	30	50	42	15	35	39	35	31	39	30
Neutral	8	15	4	12	35	66	19	23	19	35	15	27
Strongly Disagree	-	4	4	4	8	15	-	8	12	19	4	20
Strongly Disagree	-	-	-	-	-	4	-	-	4	-	-	-

Table 2: Survey results - comparing DEVS-Suite and ns-2 simulators

Aspect	DEVS-Suite	ns-2
Relative ease of use	62	38
Model libraries	30	70
Visualization	73	27
Execution Performance	65	35
Documentation	15	85

At the end of the fall semester 2009, an *anonymous evaluation* of the surveys was completed by 26 out of 30 students. The anonymity of the survey allows students to objectively evaluate the DEVS-Suite and ns-2 simulator. A five-point Likert scale (Likert 1932) with values ranging from Strongly Agree to Strongly Disagree was used. The results of their responses are shown in Tables 1 and 2. All entries in the tables represent percentages. The individual evaluation of the questions in Table 1 are self explanatory.

Considering the DEVS-Suite simulator by itself, 86% of the surveyed students found the simulator's visualization and model layout to be well-suited to their needs (see Table 1). This is according to a cumulative percentage of the responses to Q1, Q2, and Q4. The remaining 12% and 2% of the students were neutral and unsatisfied, respectively. This is consistent with student responses shown in Table 2 where 73% and 62% of the respondents prefer the DEVS-Suite simulator's visualization and ease of use over ns-2, respectively. Similarly, 75% of the respondents to Q3 and Q5 were very satisfied or satisfied with the DEVS-Suite simulator's execution performance, animation, and time-based trajectory viewing. Again, Table 2 shows the DEVS-Suite simulator offers relatively better performance as compared with ns-2. In terms of Q6, students were mainly neutral. This can be attributed to not needing to use DEVS-Suite simulator's Enable Governor (i.e., synchronizing the speed of the animation and time-based trajectory plotting) which slows down the simulation execution and inadequate documentation.

The survey results shown in Table 1 show that 81% of the students were very satisfied or satisfied with the support provided for OSPF network simulation modeling (see responses to Q7 and Q11). Responses to Q8 and Q9 suggest 67% of the students found the tool useful for model development and simulation experimentation. The remaining 21% and 2% were neutral and dissatisfied, respectively. The results of the remaining survey questions (i.e., Q10 and Q12) are not surprising since on the one hand detecting errors is usually non-trivial and the assignments used for the surveys were developed for the first time.

Considering the overall percentages for the DEVS-Suite and ns-2 simulators, the latter is preferred assuming equal weight for all the entries in Table 2. The preference for ns-2 over DEVS-Suite is not surprising since the former offers many protocol implementations belonging to seven segments of the networks such as TCP, AODV, FTP, etc. Furthermore, ns-2 has a large user base spanning several years with many contributors. The difference between the documentation for ns-2 and DEVS-Suite simulator is surprising. The ns-2 offers more in terms of user-guide and technical manuals as well as forums compared with DEVS-Suite. In comparison, textbooks and other publications on DEVS describe M&S principles, theory, and methods that are the basis for developing model specifications and simulation. The significant gap between the evaluations suggests that most students did not find the DEVS documentations accessible as it is the case with the use of formal methods in computer science. Students seem to prefer elaborate user-guides, manuals, and user-community support. These shortcoming lend support for further development for model libraries and more detailed user-guide documentation for DEVS-Suite.

In our course activities, we did an examination of the emerging ns-3 discrete-event network simulator which is intended to replace the popular ns-2 simulator. The latest version of ns-3 has been under development since 2006 and release 3.7 is tentatively scheduled for January 2010 (ns 3 2010). Based on basic experiments in classroom context, ns-3 as compared with ns-2 offers the important improvements listed below:

- **Design and implementation**: Scalability, modularity, coding style, and documentation are improved. The simulator kernel and its auxiliary components such as generators and tracers are written in pure C++. This has strengthen the object-orientation and thus helps with managing model implementation complexity.

The simulator can also work with Python scripting interface (instead of OTcl) for simulation configuration (Henderson, Lacage, and Riley 2008). Students prefer Python over OTcl since it is simpler to learn and use.

- **Performance**: ns-3 provides better performance; in particular, large models execute faster.
- **Deployment**: ns-3 is simpler to install. Its installation is supported with build tools such as waf and Python. Installation time is less than ns-2 and requires less user interaction. After installation, network models written in C++ files are easy to build and run.
- **Visualization**: ns-3 visualization capability remains limited, visualization tools are under development . For example, Gustavo Carneiro is developing the ns-3 PyViz (ns 3 2010).

7 CONCLUSIONS

Use of simulation is common across many scientific and engineering fields including computer networks. Our experience and those of others had revealed limitations of popular network simulators for education purposes. That is despite major efforts to fulfill this need with tools that are targeted for research and/or commercial applications. Given the advantages of the general-purpose DEVS-Suite simulator, we extended it with a library of computer network model components. The goal was to have a simulator that can better support teaching and learning of computer network simulation such as OSPF protocol. To evaluate the DEVS-Suite simulator, graduate students of computer network course are asked to use it and the ns-2 simulator in three homework assignments. A collection of questions focusing on both domain-neutral and domain-specific aspects of the simulators. The students' evaluations and our analysis of their answers are expected to show the extent to which each of these tools can support computer network protocol education. The results of this study should help further development of network simulator tools may also lead to development of new kinds of simulators for computer networks and possibly other types of complex systems.

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 and Arizona State University.

REFERENCES

ACIMS 2010. DEVSJAVA modeling and simulation tool. http://www.acims.arizona.edu/SOFTWARE.

Begg, L., W. Liu, K. Pawlikowski, S. Perera, and H. Sirisena. 2006, February. Survey of simulators of next generation networks for studying service availability and resilience. Technical report, Department of Computer Science and Software Engineering University of Canterbury, Christchurch, New Zealand.

Chen, Y., and H. S. Sarjoughian. 2010. A component-based simulator for MIPS32 processors. *Transactions of the Society for Computer Simulation International*. In Press.

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

Cowie, J., A. Ogielski, and D. Nicol. 2002. The SSFNet network simulator. http://www.ssfnet.org/homePage.html, Renesys Corporation.

DEVS 2010. DEVS-Suite. http://sourceforge.net/projects/devs-suitesim/.

Dijkstra, E. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1:269–271.

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

Floyd, S., and V. Paxson. 2001, August. Difficulties in simulating the internet. *IEEE-ACM Transactions on Networking* 9 (4): 392–403. http://www.icir.org/floyd/papers.html.

Fujimoto, R., W. Lunceford, E. H. Page, and A. Uhrmacher. 2002. Grand challenges for modelling and simulation. `http://www.dagstuhl.de/Reports/02351.pdf`.

Guo, J., W. Xiang, and S. Wang. 2007. Reinforce networking theory with opnet simulation. *JITE* 6:215–226.

Helser, E. 2009. Design and analysis of view syncrhonization in DEVS-Suite. Master's thesis, Computer Science and Engineering Department, Arizona State University.

Henderson, T. R., M. Lacage, and G. F. Riley. 2008, August. Network simulations with the ns-3 simulator. In *SIGCOMM*. Seattle, Washington, USA.

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

Lessmann, J., P. Janacik, L. Lachev, and D. Orfanus. 2008. Comparative study of wireless network simulators. In *ICN '08: Proceedings of the Seventh International Conference on Networking*, 517–523. Washington, DC, USA: IEEE Computer Society.

Likert, R. 1932. A technique for the measurement of attitudes. *Archives of Psychology* 140:155.

Mather, J. 2003. The devsjava simulation viewer: A modular gui that visualizes the structure and behavior of hierarchical devs models. Master's thesis, University of Arizona.

ns 3 2010. The ns-3 network simulator. http://www.nsnam.org/.

OPNET 2010. Opnet simulator. http://www.opnet.com/.

Riley, G. F., R. M. Fujimoto, and M. H. Ammar. 1999. A generic framework for parallelization of network simulations. In *MASCOTS*, 128–135.

Robertson, N. H., and T. D. Perera. 2002. Automated data collection for simulation? *Simulation Practice and Theory* 9 (6-8): 349–364.

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

Sarjoughian, H. 2010. DEVS-Suite WebStart. http://acims1.eas.asu.edu/WebStarts/.

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

Simulator, N. 2010. ns-2 network simulator. http://www.isis.edu/nsnam/ns/.

Steenstrup, M. 1995. *Routing in communications network*. Prentice-Hall.

Varga, A. 2010. The OMNeT++ discrete event simulation system. http://www.omnetpp.org/.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation*. New York: Academic Press.

Zeng, X., R. Bagrodia, and M. Gerla. 1998. Glomosim: A library for the parallel simulation of large scale wireless networks. In *In Proceedings of Parallel and Distributed Simulation Conference*, 154.

Zengin, A., and H. Sarjoughian. 2009, July. Teaching and training of network protocols with devs-suite. In *International Symposium on Performance Evaluation of Computer & Telecommunication Systems(SPECTS 2009)*, Volume 41, 104–111.

Zengin, A., H. S. Sarjoughian, and H. Ekiz. 2004. Honeybee inspired discrete event network modeling. In *16th European Simulation Symposium*, 176–182. Budapest, Hungary.

AUTHOR BIOGRAPHIES

AHMET ZENGIN is Assistant Professor in the Department of Computer Science Engineering, Sakarya University. Faculty of Technology. He was with the Arizona Center for Integrative Modeling and Simulation (ACIMS) at Arizona Stte University. His research interests are modeling and simulation of network systems, routing protocol design and implementation and swarm intelligence applications. His email address is <azengin@sakarya.edu.tr>.

HESSAM SARJOUGHIAN is Associate Professor of Computer Science & Engineering at Arizona State University and Co-Director of the Arizona Center for Integrative Modeling and Simulation. His research focuses on multi-formalism modeling, co-design, collaborative modeling, and SOC simulation. He can be contacted at <hss@asu.edu>.