



A new DEVS-based network simulator for online routing protocol education

Ahmet Zengin¹ · Cevat Altunkaya¹ · Şahin Kara² · Hessam Sarjoughian³

Received: 31 March 2022 / Revised: 19 October 2023 / Accepted: 13 December 2023 /

Published online: 2 January 2024

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

Simulation continues to be highly attractive and indispensable for studying networked systems. There remains the need for simulators that can support efficient execution in large-scale, complex models and can offer a theoretical underpinning with model abstractions. A qualified and effective simulator is still hard to find in education and research. To obtain a better solution in education and research, a collection of the parallel DEVS-based computer network model components is presented in this article. The Discrete Event System Specification (DEVS) is a mathematical modeling formalism for describing (discrete and continuous) dynamical systems. Among discrete event modeling approaches, it is well suited for formally describing concurrent processing and the event-driven nature of arbitrary configuration of nodes and links forming network systems. The suggested system is developed by emphasizing the Open Shortest Path First (OSPF) protocol. These expressive models are realized in the DEVS-Suite simulator as DEVS-Suite provides a robust environment with rich support for designing experiments and visualization. Visualization representation of a typical routing protocol logic with routing tables and control packet provides better support for active learning, particularly for distance education in COVID-19 disease. The resulting OSPF-DEVS simulation environment offers a model abstraction well-suited to educational environments. An exemplary computer network model is developed and evaluated in both simulators. The simulation models have proved very helpful in understanding the basics of the computer network's principles, designs, routing protocols, etc.

Keywords Computer networks · Protocol education · Modeling and simulation · DEVS-suite · Network simulator-2

1 Introduction

In the past few decades, network architectures and communication protocols, in particular, the Internet and its associated technologies, have become important disciplines in the research of

✉ Ahmet Zengin
azengin@sakarya.edu.tr

Extended author information available on the last page of the article

computer science engineering. While researchers are required to understand both theoretical and practical aspects of the Internet architectures and protocols, as academicians, we not only to deal with questions such as what the theory behind the network is, what its infrastructure is, how it works and how to develop it, but also to give the effort to find alternative teaching methods [1, 2] for graduate or undergraduate education.

Many materials and textbooks are available for computer network education, and theory with practice is usually combined in a classroom setting [3, 4]. Because of the complexity and scalability of the network systems, giving theory and understanding of how things work in practice on a single course is an important challenge. Modeling networks in a network course education is a technique for learning theoretical aspects of networks and communication protocols [5, 6]. Therefore, understanding the underlying concepts, principles, and theories of computer networks can significantly benefit from simulation modeling. Although various classical and contemporary approaches have been developed, there remains the need for simulators that not only can support the efficient execution of large-scale, complex models but also have a theoretical underpinning with model abstractions that can simplify the development of the network system simulations.

When making a network system or technology easy to use and understand, an appropriate abstraction or simplification approach is needed. Understanding complexity in computer networks require the ability to extract simplicity. Extracting simplicity requires a system theoretic approach, language or modeling formalism, which provides flexible mechanisms for a simplified abstract view of complex systems. For example, modern programming languages such as Java and Python have more abstractions than classic languages, making it possible to focus on higher-level actions when writing programs. Abstractions divide problems into tractable and easy-to-understand pieces, which make the task of learning easier.

While various open-source and commercial tools are available for network education, there remains a desire for simulators to support student learning and instructor teaching better. A good simulator is required to ensure a proper understanding of the technology [7]. In network protocol education, there are various open-source and commercial tools. Numerous tools (e.g., GNS3 [8], Cisco Packet Tracer [9], ns-2 [10], ns-3 [11], OPNET [12], GloMoSim [13], SSFNet [14], and OMNeT++ [15]) have been devised to satisfy the necessities of researchers. However, these tools have some disadvantages in education and training because they are primarily targeted for research and engineering. Such simulation models generally provide the basic support for (textual and limited visual) manipulation of the simulation model development and experiment. A key emphasis has been enabling the design and testing of routing algorithms, MAC layers, and end-to-end queuing. Routing protocol education requires understanding state transitions among routers, controlling message traffic and routing database creation processes. Such simulators and their visualization tools frequently ignore these routing protocol dynamics, causing limited learning of the whole routing process [16]. These tools also require expertise in the software when creating basic models [17]. Other shortcomings of the current network simulation tools are weak support for visualization, difficult deployment and usage, flexibility, scalability, performance, and web access. For example, the ns-2 is difficult to deploy, in part, due to the use of two programming languages (C++ and oTcl). The OPNET emphasizes visualization capabilities and ease of use by domain experts with a built-in library of models-consequently, a huge demand for tools strictly supporting learners in networking classes [18].

In this work, to remove network simulator shortcomings and solve problems in network education, the DEVS-Suite general-purpose simulation tool is expanded to make teaching

easier for the network routing protocols. A model repository and infrastructure for the OSPF routing protocol has been established by using software engineering principles such as the seven-layered OSI model, OSPF RFC specifications, DEVS modeling formalism, and object-oriented design principles so that the emphasis is placed on education and thus capturing the basic principles of the network protocols using sound modeling and simulation principles instead of supporting highly detailed network protocol simulations. Because the underlying DEVS formalism allows the modeler to design with an appropriate level of abstraction determined by objectives, the emerging model has a higher abstraction level and modularity.

The DEVS-Suite user interface provides a consistent, efficient, and integrated hierarchical component-based representation of models with run-time I/O and state trajectories and tabular data visualization. The use and pedagogical effectiveness of the DEVS-Suite network simulator are implemented in a classroom setting. With this simulator, students can create arbitrary network models, experiment with the models, and track their dynamics using complementary views. They can be empowered to learn the network concepts interactively.

Evaluation is also carried out for educational needs. The simulator can also be used in a standalone fashion and web-based via DEVS-Suite Web Start [19], which empowers e-learning using Java technology. Formal V&V (verification and validation) experiments of the developed model under the DEVS-Suite simulator are presented in [20], its large-scale performance capabilities in [21], and wireless models [22].

2 OSPF-DEVS network modeling approach

2.1 Donating the DEVS-Suite with networking services and applications

To assist teaching, complex dynamics of the routing protocols, modeling and simulation methodologies, and tools are predominantly used. The general-purpose nature of the DEVS modeling formalism and the DEVS-suite simulator needs to be extended with computer network concepts and methods. The simulatable models are derived from the atomic and coupled DEVS, whereas the non-simulatable models are based on (un-timed) Object-oriented models (see Fig. 1). Examples of these categories are the Node and Network classes, which are extended from the ViewableAtomic and ViewableDigraph classes, and the NETPacket class, which is extended from the message class (see Fig. 2). Next, the main network model components for developing and simulating network models are described.

A Node in the network model represents an intelligent and autonomous unit that can process packets and reflect its network knowledge. The Node model can route messages to their destinations (see Fig. 3). The link characterization is used to define the Duplex Link model (from now the name Link will be used for simplicity). It supports the representation of bidirectional communication. Since a Duplex Link model extends a parallel DEVS atomic model (see Fig. 4), it also affords simultaneous traffic in opposing directions. The state diagrams of the node and links are given in Fig. 5. For example, when the Node is in phase “queuing,” the queue is full, and a packet is received, it will enter the state “congested,” and the packet is dropped. In the case of not receiving any packets from other Nodes, if the Node completes processing a packet and its queue is not empty, it starts iteratively processing stored packets until the queue is empty. If the Node receives a packet at the same time that an internal transition is to occur, it simply completes the internal transition, and later, the packet is queued. If the state is “flooding,” the Node generates and sends LSA packets for its neighbors.

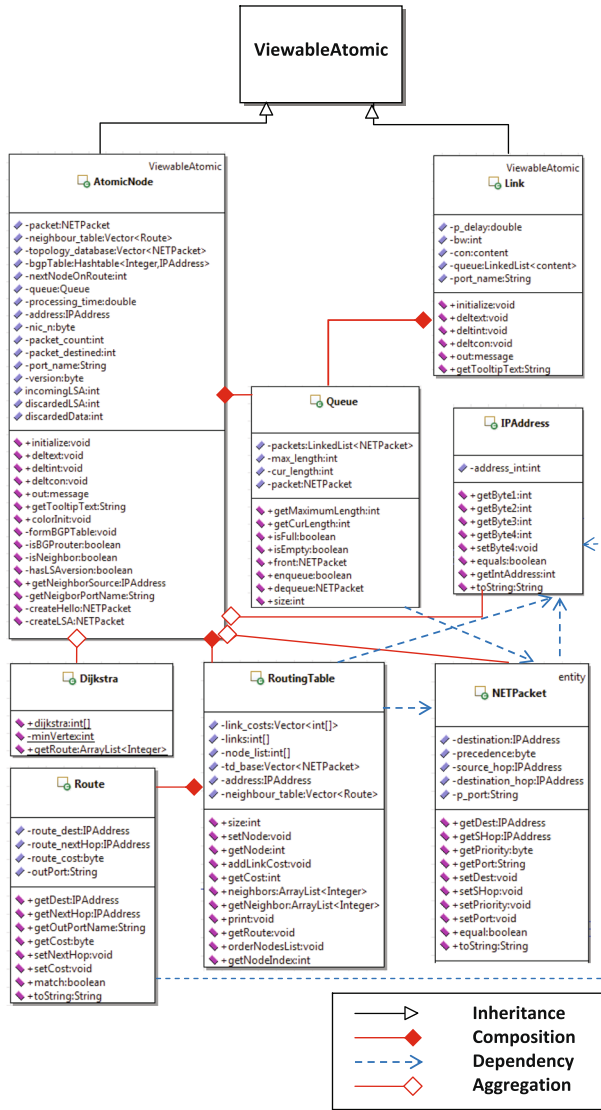


Fig. 1 OSPF-DEVS simulatable and non-simulatable classes and their structural relationships. A UML diagram is automatically produced under IDE to show the class structure of the OSPF-DEVS model. This design shows the derivation, composition and aggregation relationships as defined. The aggregation relationship between the AtomicNode, the IPAddress and Dijkstra classes can be changed to a composition relationship and, therefore, require the lifetime of the IPAddress and Dijkstra objects to be under the control of the lifetime of the AtomicNode object

2.2 Model control options during simulation experiments

Along with the Tracking Environment, DEVS-Suite provides complementary mechanisms to control model build-up examples. Figure 6 depicts the simulation time presented at the bottom. The figure shows that a slider controller assists time advance speed during the simulation.

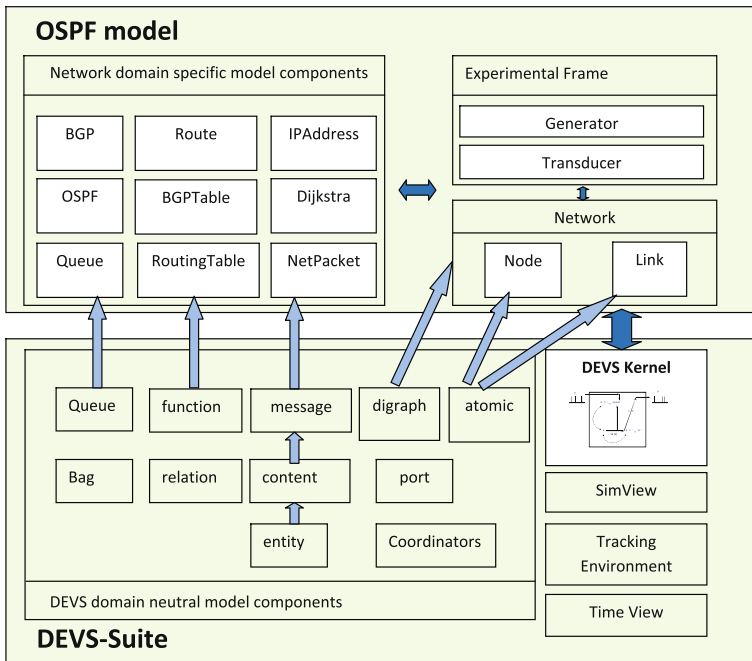


Fig. 2 A conceptual view of the DEVS-Suite with computer network model components. An OSPF model is designed on top of DEVS-Suite library classes. NETPacket class is developed by extending the Entity class. Node, Link, Generator and Transducer classes are extended from the DEVS Atomic class, and all coupled classes are derived from digraph

Model runs can be handled using Run, Step, Step (n), Request Pause, and Reset buttons, as shown in Fig. 6.

2.3 Exploiting simulation data

Data collection is the most crucial process in any simulation research [23]. DEVS-Suite supports the selection, processing and alternative observing datasets used for model simulation activity.

As shown in Fig. 6, the modeler can follow events during the simulation experiment and data for numerous models. The data from the simulation may also be gained in tabular format as an MS Excel file using a tracking logger.

2.4 Envision OSPF routing logic

It is significant to visually monitor the operation of a network protocol when educating on computer networking fundamentals and theories (see Fig. 6). Learners should learn step-by-step the logic and behavior behind the routing protocols. Although ns-2 offers a detail of granularity in the nam tool for visualization, it is impossible to master most of the protocol’s behavior through simulation visualization. Developed tool in the simview environment enables learners to understand the formation of protocol run and behavior logic of the network protocols. OSPF protocol algorithmic backgrounds are detailed in [24].

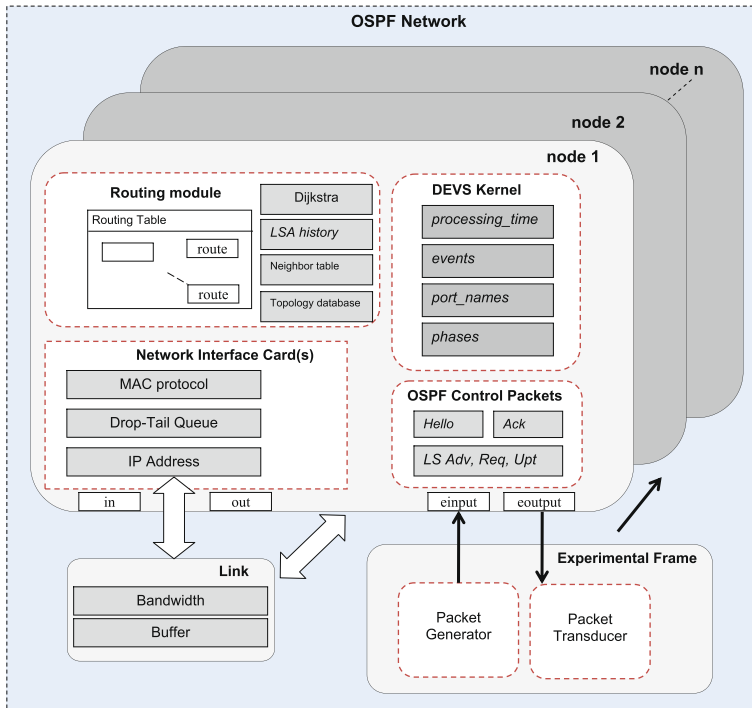


Fig. 3 The model components comprise the network and data link layers in OSPF-DEVS. Key components of a node are the routing module, network interfaces and DEVS Engine. A typical node includes packet definitions for creating and forwarding them. A node also has several data structures, such as a neighbor table and topology database for supporting data forwarding

2.5 Initialization, network discovery and database synchronization

Every AtomicNode can discover the set of other nodes in the network domain. To do this, every Node object (an instance of the AtomicNode class) calls the createHello() method and the output function (λ), after which Hello messages are broadcasted to all of its immediate neighbors (see Fig. 7). After discovering all the neighbors, every node object invokes its createLSA() method to send its topological database to all its neighbors. A Vector-typed topology database field of the Node stores the LSA packets from all other routers, and the neighbor table field stores the neighbor ID and associated interface names: the Neighbor Table and the topology databases implemented as Vector. In contrast, the RoutingTable stores the Route objects.

2.6 Shortest path calculation using the Dijkstra algorithm

The Dijkstra algorithm [25] is recognized as the shortest path algorithm. Our implementation uses it to find the shortest paths from the source to all other destinations in a weighted, directed graph. All weights are non-negative, and hop counts are selected as weights. Key methods and fields of the Dijkstra class are implemented with static modifiers, and therefore, its getRoute() method can be invoked with parameters the RoutingTable and address of the originating node unless constructing the Dijkstra class. This method returns an ArrayList storing the whole

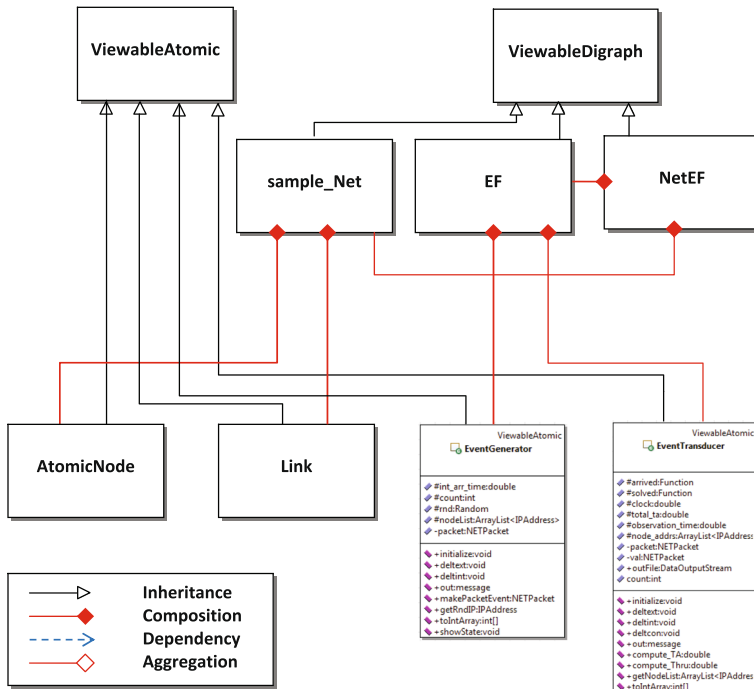


Fig. 4 OSPF-DEVS coupled model classes. The EventGenerator and EventTransducer classes are inherited from the ViewableAtomic model, and sampleNet, EF, and NetEF classes are inherited from the ViewableDigraph coupled model

path from the root to the target nodes. The Dijkstra class also generates the topology matrix and shortest path tree, as depicted in Fig. 6.

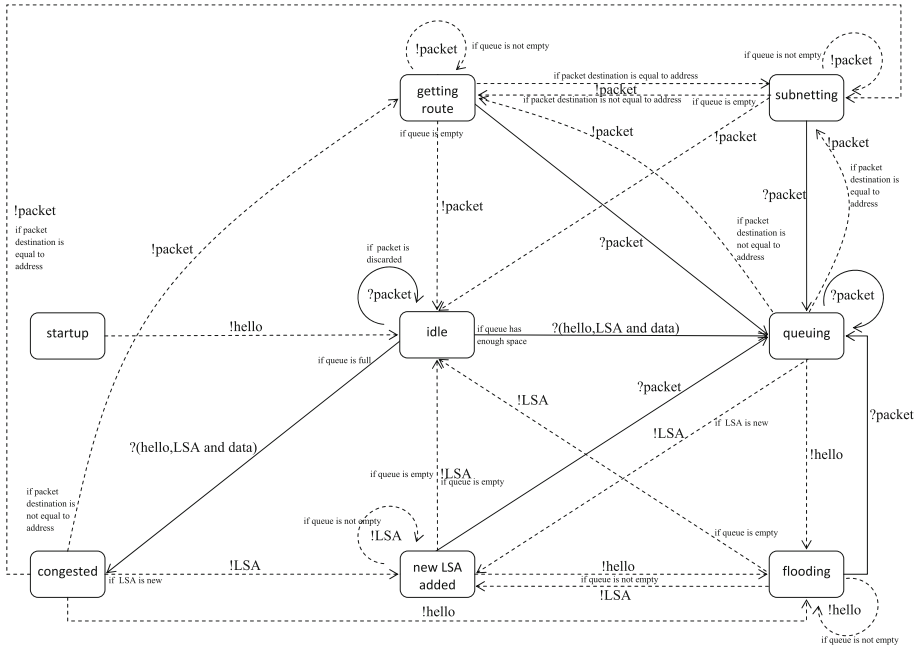
2.7 Timeview model tracking options

The TimeView facility of the DEVS-Suite has been developed for runtime drawing of datasets in the form of rich content graphs. It provides a view of developed model components’ behavioral mechanics and assets. As depicted in Fig. 8, the modeler can view input events, outputs, states and trajectories related to time-based entries under TimeView. Several behavioral views can be understood in Fig. 9. With the log databases, these charts are effective for investigating the processes over time for the models.

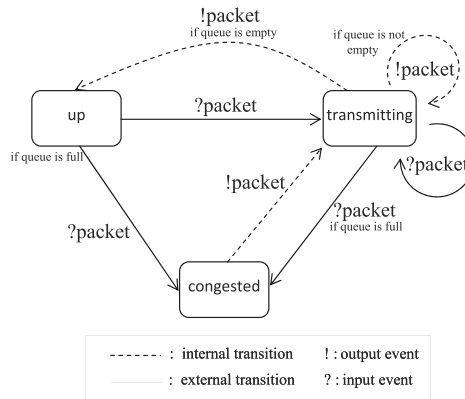
3 Assessment for engineering education

To assess the performance of the developed network simulator under DEVS-Suite, a case example is developed and applied to the Design and Simulation of Computer Networks course. The course is a graduate-level course in the Faculty of Technology, Sakarya University, Turkey. Thirty students have participated in that course.

Graduate students who taken that course are expected to use developed simulators in DEVS-Suite and ns-2 simulators for their three homework assignments. The first task or



(a) Node's state diagram



(b) Link's state diagram

Fig. 5 Node and Link's states, state transitions and output. Arrows represent internal and external transitions between node and link states. The whole chart completely represents the model behavior. For example, when the Node is in phase “queuing”, the queue is full, and a packet is received (δ_{ext}), it will enter the state “congested,” and the packet is dropped

assignment was to assess the installation of both simulators. In the first week of the term, all students must install the ns-2 simulator package on Cygwin under a Windows machine with the Network Animator (nam) visualization tool. Furthermore, a jar file is provided to them in flash memory. Then, learners were asked to assess, examine and analyze the distribution packages for both simulators.

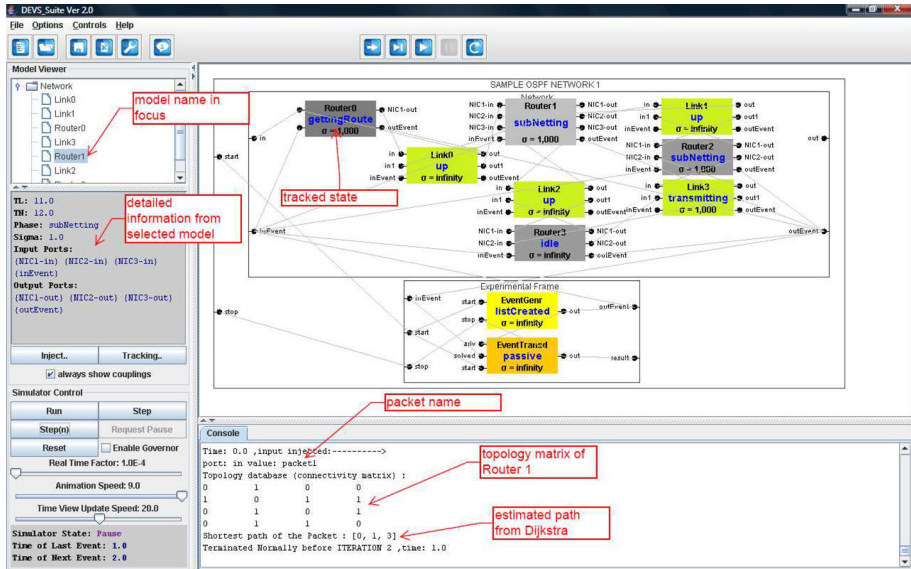


Fig. 6 The SimView viewer and an example network model with OSPF. Students can track all simulation steps in this window with generated data structures. Students can also track a network protocol’s execution visually in teaching computer network concepts and theories. Students can learn the logic behind and behavior of the routing protocols in step by step manner. The node model accommodates vector data objects for two data sets by which these objects pass to the Dijkstra class as parameters. Therefore, the shortest path is generated, as seen in the console view

To assess simulator tools for education, it is developed another assignment project. In the 2nd assignment project, participants created a network topology. This topology is formed with ten nodes and 11 bi-directional links. Network sizes are kept small because the main idea is to evaluate the protocol logic and its concepts. Though both simulators have structural differences, simulation parameters are selected the same or near. Then, students were asked to monitor the tools and note the important points. As mentioned, DEVS-Suite and the ns-2 simulators have visualization capabilities to show model behavior details such as network protocols. Not only visual tracking but also students are asked to analyze the reporting and resulting outcomes from both simulators.

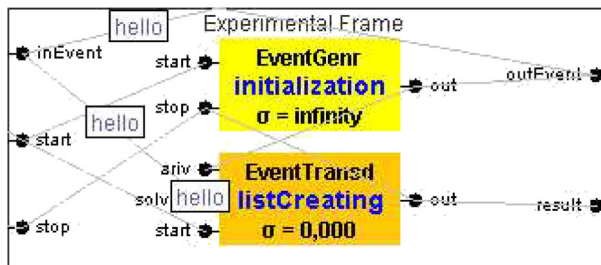


Fig. 7 Experimental Frame components with animation of messages. Routing protocol education requires understanding state transitions among routers, control message traffic and routing database creation processes. Hello messages are broadcast to all its immediate neighbors and the experimental frame for tracking purposes

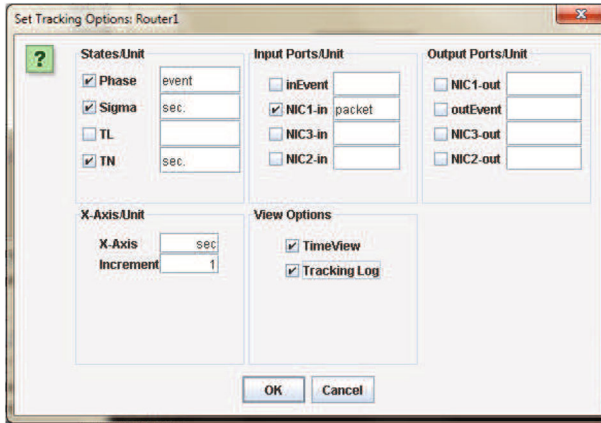


Fig. 8 Tracking window for network model. Modelers can select animation and tracking view options for any number of atomic/coupled models. All variables for a specific model can be seen graphically by selecting the appropriate selection in the window with the unit. For example, for the Router1 atomic model, network interface one input and model states are visually displayed

The last assignment is about scalability evaluation as opposed to assignment 2. In this task, students have developed very large-scale network models. To justify the evaluation of the DEVS-Suite and ns-2 tools, a recursive algorithm for creating a grid topology is used instead of using topology generator tools. The recursive algorithm is coded using Java and oTcl programming languages. Large simulation models are from a few thousand nodes to ten thousand nodes. These models are simulated across varying conditions and experimental

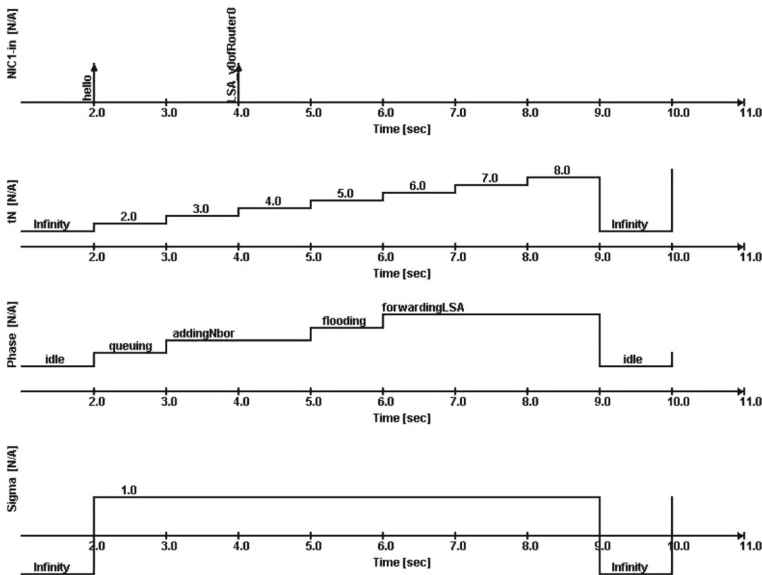


Fig. 9 Trajectories of OSPF protocol model. These charts and log files are useful in examining the time-based operations of the OSPF protocol simulation components

settings, observed the behaviors, and collected and analyzed the results. Consequently, the effectiveness as an educational tool is evaluated, and the strengths/weaknesses of both tools are reported.

After these three assignments, a survey is applied to students taking that course. The survey is organized into three parts. The twelve questions are asked of the participants. The questionnaire/survey is divided into Sections I, II and III. Section I assessed the developed simulator's visualization and animation capability (questions from 1 to 6) (see Fig. 10). Section II assessed the degree of support to teach routing protocols, network structure and behavior.

In Section III, students are asked for a comparison of both simulators (see Fig. 11). The queries are dedicated to simplicity of use and evaluating the effectiveness and speed of learning and using these tools. Other important evaluation factors were the presence of model libraries, visualization support, execution performance, and documentation access. All questions under sections can be found in [26].

After the semester, a survey from 26 out of 30 students is conducted as an anonymous assessment. The secrecy of the survey permits students to assess both simulators accurately. The five-way Likert scale [27] was used. This scale is varying from Strongly Agree to Strongly Disagree. The outcomes of students' responses are presented in Figs. 10 and 11 representing the percentages.

According to the survey results presented in Fig. 10, 86% of the students reported positive feedback about DEVS-Suite's visualization capability to track model behavior in terms of their educational needs. Q1, Q2 and Q4 questions generate the results. 12% of the students are impartial, and just 2% of the students are unfulfilled. These results are consistent with the values in Fig. 11. According to Fig. 11, 73% of the students preferred visualization properties of the DEVS-Suite simulator. 62% of the participants also rated the developed simulator's ease of use, while ns-2 is preferred with 38%. Execution performance, animation, and time-

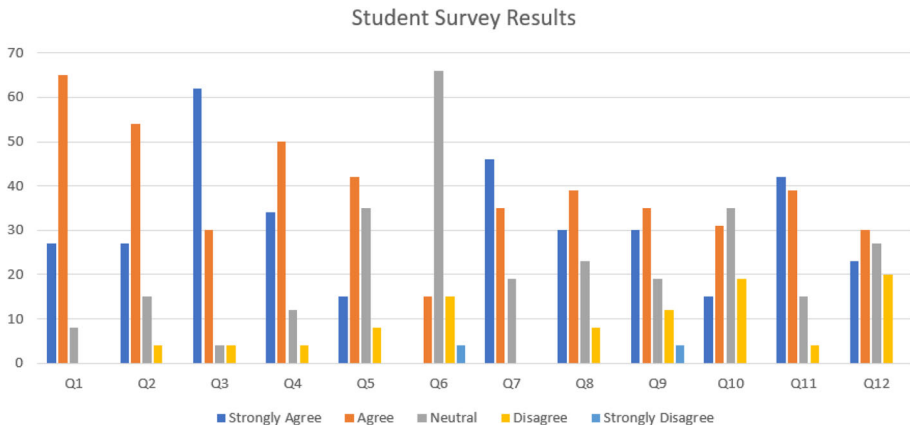


Fig. 10 Student survey results. Eighty-six percent of the surveyed students found the simulator's visualization and model layout well-suited to their needs. This is according to a cumulative percentage of the Q1, Q2, and Q4 responses. The remaining 12% and 2% of the students were neutral and unsatisfied, respectively. 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. Responses to Q8 and Q9 suggest that 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

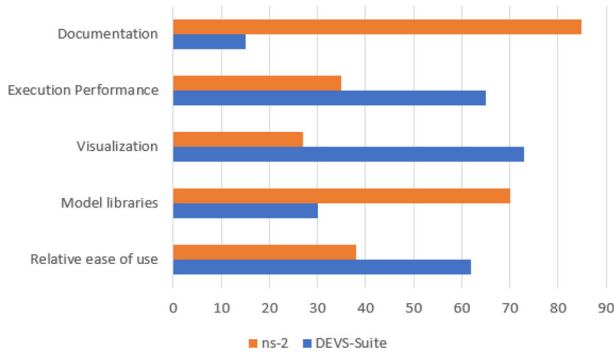


Fig. 11 DEVS-Suite and ns-2 comparison survey results. The ns-2 offers more in terms of user guides and technical manuals as well as forums compared with the DEVS-Suite. In comparison, textbooks and other publications on DEVS describe M&S principles, theories, and methods that are the basis for developing model specifications and simulation. The significant gap between the evaluations suggests that most students did not find the DEVS documentation accessible, as with the use of formal methods in computer science. Students seem to prefer elaborate user guides, manuals, and user-community support. These shortcomings support the further development of model libraries and more detailed user-guide documentation for the DEVS-Suite

based trajectories of the DEVS-Suite simulator are measured with questions Q3 and Q5. Results showed that 75% of the respondents are fulfilled by the developed tool. As a result, Fig. 11 proves the developed simulator's performance compared to ns-2.

On the other hand, the answers to Q6 are a little surprising, and 66% of the students are neutral. This is because the Enable Governor property of the DEVS-Suite is not considered a required function by students during their training activity.

According to the results in Fig. 10, simulation support for the developed tool is measured by 81% of students being very satisfied (see Q7 and Q11 responses).

According to Fig. 11, percentages of the preferences for both simulators are presented. Concerning the results, the ns-2 network simulator seems preferred when considering all entries. Unsurprisingly, ns-2 is preferred over DEVS-Suite, as the former offers many protocol implementations, such as TCP, AODV, FTP, etc., belonging to the seven segments of networks. Also, ns-2 has had a huge user and developer community over many years since developed time. Therefore, it has good documentation and a library.

4 Discussion

4.1 Event frequency settings

The wall clock execution times for the simulations vary significantly. The minimum, average, and maximum wall-clock execution times of sample four-node network models for ten replications are 11, 13.8, and 18 seconds for the DEVS-Suite and 50, 58, and 70 seconds for the ns-2 simulator. The difference between simulation execution times comes from the abstraction levels of both simulators. It is clear that the DEVS-Suite simulator allows the modeler to focus on higher levels rather than dealing with unnecessary details, i.e., there is no need to emphasize link layers in protocol education.

In addition, another difference between the DEVS-Suite and ns-2 approaches is that in the former, adjustments can be performed during simulation experiments. At the same time, in the latter, the number of events and the event tracking scheme cannot be altered (see Fig. 12).

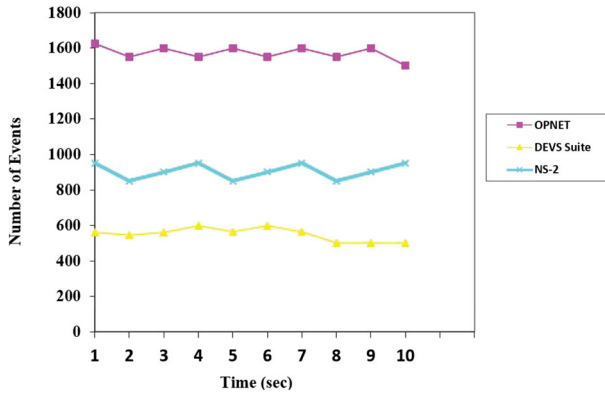


Fig. 12 Event frequency of the selected simulators per time unit. Trace data obtained from experiments in ns-2 may be huge and difficult, or even impractical, to process by overflowing the limits of spreadsheet tools. This kind of tracking decreases the performance by generating simple statistical data. In contrast, the DEVS-Suite enables the modeler to choose between the number of events and/or tracking data and the desired simulation performance

4.2 Visualization

The three complementary views-hierarchical component structure and message animation, time-based trajectories, and tree listing of the model - provide important capabilities for students and instructors to view the structural and detailed aspects of the simulation models. The routing protocol can be examined and displayed step-by-step through visual animation of nodes and delivery of messages. User-triggered inputs, outputs, and previously specified state variables can be drawn by time graphs and trajectories. Figure 6 shows four routers and four links network models with an experimental frame. Simulated models and their predefined states, input and output ports (Network Interface Cards - NIC), and their couplings, as well as the transmission of the packets, can be seen. The tooltip view provides user-defined detailed information for each atomic model (see Fig. 13). Also, users can examine the composition hierarchy structure of models.

4.3 Expertise

Two key requirements to use many simulation tools are to be an expert in at least one programming language and be a mature software developer. For this reason, some tools (e.g., OPNET) are developed for domain experts who need more software development experience. Other tools (e.g., ns-2) require maturity in software design, code development, and testing. This is because scripting languages hide important details about the structure and behavior of models. If one is to synthesize simulation models at the abstraction levels supported by ns-2, using a tool such as OPNET may be advantageous. Other tools such as DEVS-Suite strike a balance between simulators such as ns-2 and OPNET [12]. While restrictive, the strict hierarchical basis of the DEVS formalism is advantageous since it simplifies model development compared to the modeling approaches that require detailed knowledge and experience with object-oriented programming languages. Another benefit of the DEVS-Suite simulator is its pure implementation in the JavaTM programming language.

Fig. 13 Tool tip View of a Router. The entities belonging to network systems, such as routes, routing tables, packets and queues, can also be visually tracked by a developed simulator. If the mouse is placed on a network node, a context menu shows the neighbors table and routing database of related nodes with all user-defined model states selected for viewing

```

IPAddress: 0.0.0.1
class: swarmNet.Framework.Node
phase: idle
sigma: infinity
tL: 12,000
tN: ∞
QueueState: 0 packets(0KB)
Neighbour Table :
#ofNeighbour: 3
[dest:0.0.0.2 nextHop:0.0.0.2 Out Port:NIC2-in cost:1
, dest:0.0.0.0 nextHop:0.0.0.0 Out Port:NIC1-in cost:1
, dest:0.0.0.3 nextHop:0.0.0.3 Out Port:NIC3-in cost:1
]
Topology Database:
#ofLSA's: 3
[LSA_v0ofRouter0
, LSA_v0ofRouter2
, LSA_v0ofRouter3
]
#ofPacketsRouted: 0

```

4.4 Web access

DEVS-Suite is accessible via Java Web Start Technology. The DEVS-Suite Webstart has been used in the Modeling and Simulation Theory and Application course offered to on-campus and online students at Arizona State University DEVSJAVA WebStart since spring 2006 and the DEVS-Suite WebStart in spring 2009 [19]. Given the remote accessibility and no loss of simulator performance, students can be introduced to the simulation without being required to install and configure a simulator. After students have gained a basic understanding of the simulator, they are better prepared to use it using IDEs such as Eclipse.

4.5 User-base

Since the DEVS-Suite is developed based upon an open-source software project and the DEVS user community tends to grow, we expect additional model libraries to be developed and the existing ones to be improved. Furthermore, recent domain-specific application models for simulating service-based software systems [28] and hybrid agent/cellular automata models [29] are encouraging for simulating systems beyond computer networks (see Table 1).

Table 1 A comparison between DEVS-Suite and selected simulators

Aspect	DEVS-Suite	OPNET	ns-2	OMNeT++
Visualization	high	high	low	high
Expertise	medium	low	high	high
Web access	high	low	low	low
User-base	low	high	high	medium

5 Conclusion

This paper introduces a new simulation framework for teaching, education and research of computer networks and related technologies. The concept, approach, and technical details of the OSPF-DEVS models are presented. Simulation examples and experiments were developed to show the inner workings of the simulator and compare it with other network simulators. The DEVS-Suite simulator for modeling and simulation of OSPF solves the challenges of ns-2 concerning visualization and usability. The ns-2 is generally used for protocol development and comparison for research purposes [30]. In contrast to ns-2, the developed simulator can be executed over the web by exploiting Java Web Start Technology. The DEVS-Suite WebStart version of the underlying simulation technology enables the modeler to access the latest simulation package and model libraries and execute them under browsers. In this paper, results depict that trainees are much from the opportunities of the developed simulator [26]. The simulator's visualization, tracking and automation capability for monitoring components' behavior as plots, time trajectories and tabular data assist when teaching, training and learning of routing and network theory. Visualization representation of a typical routing protocol logic with routing tables and control packet provides better support for active learning, particularly for distance education in COVID-19 disease [4]. Students can easily follow and track the creation and alteration of tables and associated databases. Students eventually can gain knowledge of M&S approaches for performance evaluation of networking experiments. Future work may include developing advanced network protocol models and their use in the classroom. Additionally, the DEVS-Suite can be extended with stronger time trajectory visualization and data analysis support.

Author Contributions All authors contributed to the study conception and design.

Funding The authors did not receive support from any organization for the submitted work.

Availability of data and materials Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.


References

1. Li Y, Lu S (2023) Retraction note: research on physical education system model using multimedia technology. *Multimedia Tools Appl* 82(13):20705–20705. <https://doi.org/10.1007/s11042-022-13864-2>
2. Yağanoğlu M, Bozkurt F, Günay FB, Kul S, Şimşek E, Öztürk G, Karaman S (2023) Design and validation of iot based smart classroom. *Multimedia Tools and Applications*. <https://doi.org/10.1007/s11042-023-15872-2>
3. Korkmaz T, Ramirez T (2006) Teaching Lab-based Computer Networking through the Designing of a Miniature Internet (MINT). <https://api.semanticscholar.org/CorpusID:16030216>
4. Patel B, Alsadoon A, Prasad PWC, Dawoud A, Rashid TA, Alsadoon OH, Jerew OD (2022) Secure data transmission in a real-time network for a tele-training education system. *Multimedia Tools and Applications* 81
5. Davis N, Ransbottom S, Hamilton D (1998) Teaching computer networks through modeling. *Ada Lett. XVIII* 5:104–110. <https://doi.org/10.1145/291712.291758>

6. Liu Y, Zhu L, Liu F (2020) Design of multimedia education network security and intrusion detection system. *Multimedia Tools Appl* 79(25):18801–18814. <https://doi.org/10.1007/s11042-020-08724-w>
7. Qiang S (2019) Formulation of physical education and training program based on multidimensional education data mining. *Clust Comput* 22(2):5017–5023. <https://doi.org/10.1007/s10586-018-2470-y>
8. GNS3 (2023) The graphical network system 3. <https://www.gns3.com/>
9. CPT (2023) The cisco packet tracer. <https://www.packettracernetwork.com>
10. Ns-2 (2023) The Ns-2 network simulator. <https://www.isi.edu/nsnam/ns/>
11. Ns-3 (2023) The Ns-3 network simulator. <http://www.nsnam.org/>
12. OPNET (2023) OPNET simulator. <http://www.opnet.com/>
13. Zeng X, Bagrodia R, Gerla M (1998) GloMoSim: a library for the parallel simulation of large scale wireless networks. In: *In Proceedings of parallel and distributed simulation conference*, p 154
14. Cowie J, Ogielski A, Nicol D (2002) The SSFNet network simulator. <http://www.ssfnet.org/homePage.html>, Renesys Corporation
15. Varga A (2023) The OMNeT++ discrete event simulation system. <http://www.omnetpp.org/>
16. Yalcin N, Altun Y, Kose U (2015) Educational material development model for teaching computer network and system management. *Comput Appl Eng Educ* 23(4):621–629. <https://doi.org/10.1002/cae.21636>
17. Marquardson J (2019) Simulation for network education: transferring networking skills between simulated to physical environments. *Inform Syst Educ J* 17:28–39
18. Kalkan ÖK, Karabulut Ş, Höke G (2021) Effect of virtual reality-based training on complex industrial assembly task performance. *Arab J Sci Eng* 46(12):12697–12708. <https://doi.org/10.1007/s13369-021-06138-w>
19. Sarjoughian H (2023) DEVS-Suite WebStart. <http://acims1.eas.asu.edu/WebStarts/>
20. Zengin A, Ozturk MM (2012) Formal verification and validation with devts-suite: OSPF case study. *Simul Model Pract Theory* 29:193–206
21. Zengin A (2010) Large-scale integrated network system simulation with devts-suite. *KSII Trans Internet Inf Syst* 4(4):452–474
22. Tuncel S, Ekiz H, Zengin A (2016) Design and implementation of a new manet simulator model for aodv simulation. *Turkish Journal of electrical engineering and computer sciences*, 2239–2254
23. Robertson NH, Perera TD (2002) Automated data collection for simulation? *Simul Pract Theory* 9(6–8):349–364
24. Zengin A, Sarjoughian H (2009) Teaching and training of network protocols with devts-suite. In: *International symposium on performance evaluation of computer & telecommunication systems(SPECTS 2009)*, vol 41, pp 104–111
25. Dijkstra EW (1959) A note on two problems in connexion with graphs. *Numer Math* 1:269–271
26. Zengin A, Sarjoughian H (2010) devtsuite simulator a tool teaching network protocols. In: *Winter simulation conference, Baltimore, Maryland, US*
27. Likert R (1932) A technique for the measurement of attitudes. *Archives of Psychology* 140:1–55
28. Sarjoughian HS, Kim S, Ramaswamy M, Yau S (2008) An SOA-DEVS modeling framework for service-oriented software system simulation. In: *Winter simulation conference, Miami, FL*, pp 845–853
29. Mayer GR, Sarjoughian HS (2009) Composable cellular automata. *Simulation: transactions of the society for modeling and simulation international*
30. Alkenani J, Nassar KA (2023) Enhance work for java based network analyzer tool used to analyze network simulator files. *Institute of advanced engineering and science* vol 29 No 2: February 2023

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Ahmet Zengin¹  · Cevat Altunkaya¹ · Şahin Kara² · Hessam Sarjoughian³

Cevat Altunkaya
cevat.altunkaya@alberen.com.tr

Şahin Kara
sahinkara@subu.edu.tr

Hessam Sarjoughian
hessam.sarjoughian@asu.edu

¹ Computer Engineering, Sakarya University, Esentepe Campus, Serdivan 54187, Sakarya, Türkiye

² Computer Programming, Sakarya Applied Science University, Kaynarca 54650, Sakarya, Türkiye

³ School of Computing and Augmented Intelligence, Arizona State University, 1151 S Forest Ave, Tempe 85281, Arizona, US