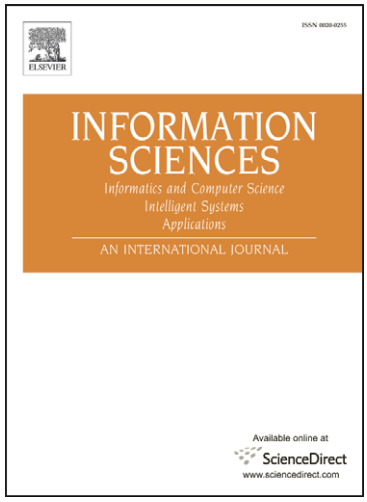


Modeling Discrete Event Scalable Network Systems

Ahmet Zengin

PII: S0020-0255(10)00522-0
DOI: 10.1016/j.ins.2010.10.023
Reference: INS 8884

To appear in: *Information Sciences*

Received Date: 15 September 2009
Revised Date: 16 October 2010
Accepted Date: 18 October 2010

Please cite this article as: A. Zengin, Modeling Discrete Event Scalable Network Systems, *Information Sciences* (2010), doi: 10.1016/j.ins.2010.10.023

Modeling Discrete Event Scalable Network Systems

Ahmet Zengin[1]

Department of Computer Science Education, Sakarya University, Turkey E-mail: azengin@sakarya.edu.tr

Abstract

Scalability in simulation tools is one of the most important traits to measure performance of software. The reason is that today's Internet is the main instance of a large-scale and highly complex system. Simulation of Internet-scale network systems has to be supported by any simulation tool. Despite this fact, many network simulators lacks support for building large models. In this work, in order to propose a new approach for scalability issue in network simulation tools, a network simulator is developed based on behavior of honeybees and high performance DEVS, modular and hierarchical system theoretic approach. A biologically-inspired discrete-event modeling approach is described for studying networks' scalability and performance traits. Since natural systems can offer important concepts for modeling network systems, key adaptive and emergent attributes of honeybees and their societal properties are incorporated into a set of simulation models that are developed using the Discrete Event System Specification approach. Large-scale network models are simulated and evaluated to show the benefits of nature-inspired network models.

Key words: DEVS Formalism, Modeling and Simulation of Networks, Routing, Scalability, Honeybees

1. Introduction

Computer based simulation is widely used in almost all areas of networking research for analyzing networked systems. Simulation is also particularly useful in allowing the network designers to test new protocols or to change the existing protocols in a controlled and reproducible manner. Treats such as modeling capability, credibility of simulation models and results, extendibility, usability and scalability should be taken into account when a new network

simulation tool is developed. There exist a number of high quality simulation tools which are widely in use, such as ns-2 [21], ns-3 [22], OpNet[23], Omnet++[37], SSFNet[8], pdns[27], GloMoSim[43] and Ptolemy project[26]. These simulation tools allow researchers and developers to test, compare and validate new and existing protocols under various conditions. These tools exploit various discrete event network methodology but have many problems. Simulators such as OPNET have detailed visualization tools, documentation and commercial devices such as routers, switches, hubs in both wired and wireless area however have disadvantages so that modeling capability is limited to a few hundred nodes in a single machine. Therefore studying larger models become a challenge due to lack of distributed execution base in simulator architecture. On the other hand, although ns-2 is fairly easy to use once you get to know the simulator, it is quite difficult for a first time user, because there are few user-friendly manuals and it is difficult to install. Various extensions and parallel and distributed variations are developed to achieve scalability. Most well-known is pdns. These tools lack system theoretic background and therefore difficult to reuse, update and extend their models.

The design of most network simulators is usually non-formalized. Lack of any formal definition causes a low performance, bad-scalable and non-reusable software architecture [6]. Formalization of the simulation design is required to design large-scale, efficient and distributed complex dynamic systems because it facilitates verification of optimal system design and efficiency [7]. In other words, because network systems commonly necessitate stage-by-stage design, verification, validation and a final integration, using a non-formalized design is not practical.

Formalized design approaches such as DEVS have many important advantages in system design. Such an approach reduces model development times and renders possible to recognize critical system design problems at earlier phases. DEVS based system theoretic model facilitates testing and improved experimentation and thus leads to higher quality models. This approach also supports design reusability by using a model repository and interoperability between simulation models. Another key advantage is that model verification, validation and accreditation can readily be done during model development. DEVS can be used as a formal tool for systems development and execution which improves model maintainability, multi-formalism modeling, automated parallel and real-time execution [42].

On the other hand, many network systems supporting inter-connectivity

are required to exhibit essential traits such as adaptability, scalability, and reliability (survivability) which are already observed in biological systems such as ants and honeybees. Large-scale biological systems, such as bee colonies, have advanced mechanisms that are scalable and adaptable under varying environmental conditions[5]. The desirable characteristics of the bee colony, scalability, adaptability and survivability, are not present in any single bee. Rather, they emerge from the collective actions and interactions of all bees in the large-scale colony. The design of complex and scalable network applications, therefore, stands to benefit from the power of biological principles and schemes.

The focus of this paper is on applying biological principles, mechanisms and DEVS formal system specification to the design and implementation of large-scale network applications. Swarm-based routing algorithms offer a number of attractive features including autonomy, robustness and fault tolerance. Distributing intelligence on the network provides rapid control over resources that can dynamically adapt to user's requirements. Such swarm-based algorithms adapt well to dynamic topologies. A new class of hierarchical routing algorithm is devised based on principles of biological swarms, which have the potential to address some of the problems in an autonomous and intelligent fashion. To develop and study dynamic and adaptive swarm-based large-scale network models as well as various routing protocols, a DEVS (Discrete Event System Specification)[42] network model is devised in DEVSJAVA [30] which is an implementation of the DEVS framework.

In this work, the nodes and links are characterized as the elementary network components. Networks with varying topologies and scales are modeled and simulator using the DEVS hierarchical model composition concept. For example, clusters are used to study its impact on reducing communication and increasing performance. The developed model is applied to comparison experiments with ns-2 to depict its credibility. The performance of the developed environment for networks having from tens to several thousands of components and connections are investigated.

The remainder of this paper, starting in Section 2, presents the State of Art. In Section 3, the modeling concepts of SwarmNet network simulator are mapped to a set of adaptable agent-based DEVS modeling constructs. The nodes and links are elaborated with supporting components. In Section 4, SwarmNet and ns-2 performance comparison as well as their evaluation are given. Section 5 is about the technics and ideas behind large models in the developed simulator. In Section 6, example models in the SwarmNet simula-

tion environment are developed and analyzed. In Section 7, evaluation of the developed environment with some features of SwarmNet network simulator is presented and some future research directions are summarized in Section 8.

2. Background And Related Work

2.1. Network Simulation Tools

Tools such as ns-2 [21], ns-3 [22], OPNET [23], OMNET++ [37], GloMoSim [43] and SSFNet [8] 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 foundations lack 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 [15]. Summary of the network simulators and their strengths and weaknesses are presented in Table 1.

2.2. Discrete Event System Specification (DEVS)

The dynamics of network systems can be described using discrete event modeling. This is because 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) [42] 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

Atomic model= (X, S, Y, δext, δint, $\delta conf$, λ, ta).

Table 1: Network simulators comparison

Aspect	Ns-2	pdns	OPNET	OMNET++	J-Sim	SSFNET	GloMoSim	SwarmNet
Object-orientation	M	M	S	M	VS	VS	M	VS
Network Models Library	S	S	S	S	M	W	M	W
Analysis of the results	M	M	VS	W	W	W	S	VS
Extendibility	M	M	S	VS	VS	VS	VS	VS
Expertise need	VS	VS	W	S	W	S	W	M
Deployment	W	W	S	M	VS	S	S	VS
Documentation	M	M	VS	S	W	W	M	M
Availability	VS	S	W	VS	VS	VS	W	VS
Visualization	W	W	S	S	M	VS	VS	S
User base	VS	W	S	S	M	W	W	W
Scalability	W	VS	M	M	S	VS	VS	VS
Performance	S	VS	M	M	S	VS	M	VS
Randomness	VS	VS	W	S	W	W	W	VS
Failure modeling	VS	VS	VS	M	W	M	W	VS
Web access	-	-	-	-	S	-	-	VS

M - medium S - strong VS - very strong W - weak

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. The output function is used to generate outputs.

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.

Computational realizations of the DEVS formalism and its associated simulation protocols are executed using simulation engines such as DEVS-Suite[19] and DEVSJAVA[1]. DEVS-Suite and DEVSJAVA are an object oriented realization of Parallel DEVS. They support describing complex structures, behaviors of network systems using object-oriented modeling techniques and advanced features of the Java programming language. The formal foundation of DEVS, its efficient execution, and the availability of sequen-

tial, parallel, or distributed simulation engines using alternative computational environments such as CORBA, HLA, and Web-services are important considerations. Furthermore, discrete event models are extended with other kinds of models such as fuzzy logic[29] [20].

`DEVS-Suite` is an open source, discrete event, general-purpose simulation environment [19]. It is a new generation extended from the DEVS-JAVA simulator and DEVS Tracking Environment. The main modules of the `DEVS-Suite` are DEVSJAVA [1], DEVS tracking Environment [31], and timeview [19]. `DEVS-Suite` can simulate models specified using the DEVS formalism [42]. The architecture of the `DEVS-Suite` simulator environment is Model Facade View Control (MFVC) [31] by which simulation data can be displayed with its animation and viewing of time trajectories generated by the parallel DEVS abstract simulator. Soft synchronization among timeviews and animation is supported based on the simulator's logical (or real-time) execution speed [16].

2.3. Swarm Intelligence-based Network Management Schemes and Honeybee Colony

Swarm intelligence (SI) is a kind of collective intelligence observed on social insects such as ants and bees[4]. SI is emerged from complex and collective intelligent behavior through interactions of autonomous swarm individuals from tens to thousands[18]. SI systems are composed of interacting locally with one another and with their environment[40]. The constituent members use very simple intelligence but resultant emerged intelligence is highly complex and collective without any central authority [34]. The social insect examples of SI are exemplified such as ant and bee colonies, bird flocking, animal herding, bacterial growth, and fish schooling[40]. SI has many appropriate properties for routing problem in distributed systems. These properties can be listed as scalability, fault tolerance, adaptation, speed, modularity, autonomy and parallelism[5]. In order to exploit these properties in network routing, some algorithms and protocols are developed by researchers such as ant colony optimization (ACO)[11], particle swarm optimization (PSO)[24] [38], mobile agents[3], [39], [32], [10] and [17]. Some works are also based on honeybees such as [36], [25], [12] and [46].

Insect societies such as honeybees have fundamental mechanisms that allow them to preserve their survivability in the presence of alterations on their environment. For example, even though nectar availability in honeybees' ecological setting may change rapidly and unpredictably, honeybees are

able to cope with such critical changes and grow their population to record numbers[33]. Honeybees' sophisticated regulation mechanisms allow them to grow very large colonies by adapting to fluctuating and ephemeral resources. Foraging behavior in honeybees is a good example for investigating social insect metaphors such as self-organization. Honeybees collectively decide selection of nectar and pollen resources and allocation of workers (foragers and scouts) to various tasks through self-organization. These selection and allocation processes among honeybees of a hive are performed in the absence of any central management authority. In a decentralized and concurrent way, each bee obeys to a set of simple rules based on some metrics such as nectar concentration, and distance and travel time to food source. These metrics, including parameters such as the number of bees responsible for storing food in the hive, determine profitability of a nectar source. If the colony encounters more than one nectar source, the most profitable one is preferred by foragers relative to other sources with less profitability. Foragers are distributed among nectar sources using profitability criterion during the course of nectar collecting process. If the amount of a nectar at a given location changes, then the importance of the nectar source is reduced for the whole colony. Furthermore, the colony deploys a relatively small number of its population called scouts to search for nectar. Scouts locate rich sources and monitor nectar availability in the environment[2]. The assignment of forager bees to food sources according to the profitability criterion is known as the scout-recruit process. One of the most well-known mathematical models of the honeybee system was developed by [33].

3. SwarmNet Network Modeling Framework

In this work, to develop a framework for modeling and research of network protocols, a set of basic network simulation model components are defined including nodes which communicate with one another via links as detailed next (see Figures 1). Framework is developed based on a previous work called SwarmNet of which background is designed using the concepts from swarm intelligence[46]. A detailed definition of its modeling and simulation concepts as well as its swarm intelligence based design can be found in [45, 44]. By coupling basic model components in DEVSJAVA, a variety of network configurations can be developed and their characteristics can be investigated (see Figure 1). Since it is assumed that only nodes and links of a network are able to cause bottleneck, they are modeled as parallel DEVS atomic models

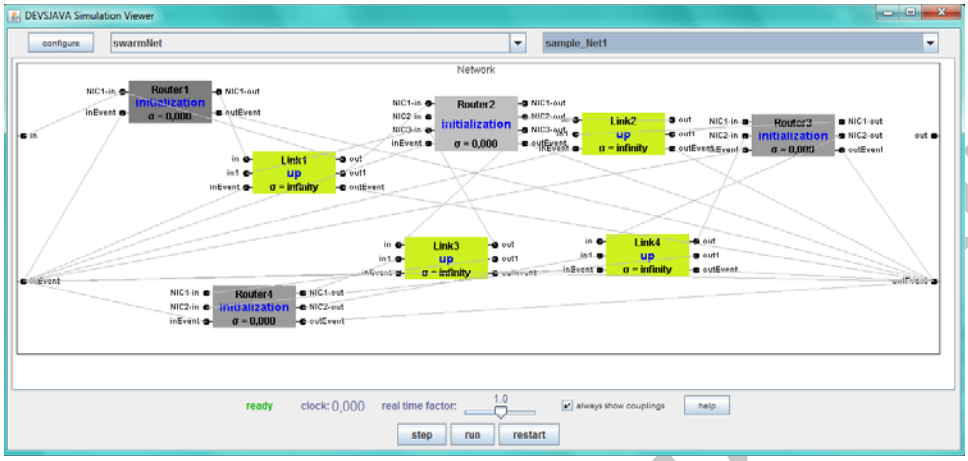


Figure 1: The model components making up the network and data link layers.

so that their dynamics are in focus and only their states as well as input and output variables are of interest. Other network components such as packets, queues and routing tables are realized and modeled as stateless entities. In the following sections these models and their associated components are defined briefly.

SwarmNet network simulator is designed to simulate of distributed system components. A distributed system is typically composed of static and dynamic parts, such as IP addresses and messages interchanging between atomic models. In this section, a brief information is given for some parts of the network system.

3.1. Basic Atomic Model Components

3.1.1. Router nodes

The poly-functional nodes and links in the network are modeled as DEVS atomic components. All nodes have several inputs and outputs through which messages among nodes can be received and sent (see Figure 1). Each input and output port pairs constitute a network interface card (NIC) which provides fundamental inter-networking services. The NIC corresponds to physical layer of the network system, but detailed MAC protocol is not modeled. A higher level of abstraction is selected due to the need for performance for large-scale experiments. An IP address as a unique id, unique name or code identifying each computer and user is assigned to every node in the network so that a packet can be directed to a specific destination. IP addresses also

specify the location of a router in the network. At each node, packets are forwarded to their destination by using information stored in its Routing Module, which defines a node's routing capability and intelligence. Also, router model is supported with a beehive. Beehive can generate and deploy any kind of control packets such as Hello, LSA and RIP messages and even artificial bees and ants for agent-based swarm intelligence applications. In SwarmNet approach a beehive is configured to launch scouts, foragers and drones to monitor and reconfigure network resources[45].

3.1.2. Border nodes

In large-scale modeling and simulation applications, in order to implement a clustering scheme for hierarchic routing, some concepts such as size of the clusters and organization of them have to be defined. By this reason, a mechanism for hierarchic large-scale routing is defined and modeled. A border node implementation is adopted for colonization or clustering. In the SwarmNet environment, border nodes are same as regular nodes and routers except border nodes have additional routing databases for distant autonomous systems and their protocols are different than regular nodes i.e., different agent bees such as drone bees are deployed for and protocol similar to BGP[35] interior gateway protocol. A routing module of a border node includes a routing table for local network as well as a global routing table which can be used to manage the routing between the autonomous systems and other parts of the global network (see Figure 2). A border node makes communications possible for outside of its autonomous system .

3.1.3. Large-scale traffic model

It is important to generate the user traffic for networks and observing how the network dynamics evolve. An event generator atomic model that can generate data packets was devised. The event generator atomic model generates packets with fixed time intervals by randomly choosing source and destination addresses. The number of packets, their lengths and frequency are determined by selected probability or traffic models such as uniform, random and Poisson. The generator model can also create and schedule specific events for the network model such as 'link down' and 'node congestion', as well as parameter variation, called bursts, to test algorithms in highly dynamic conditions.

A transducer atomic model was devised to collect and analyze the network dynamics. The observed data is stored in trace (CSV) files. The trans-

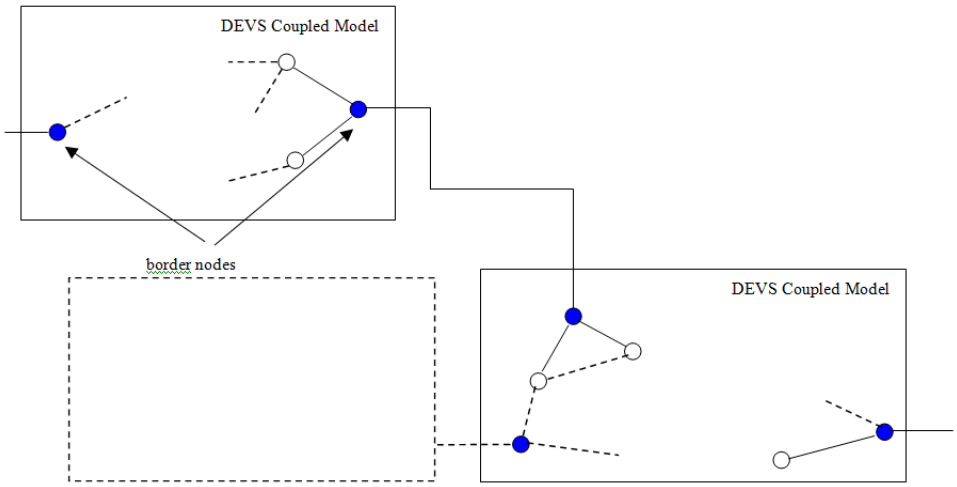


Figure 2: Border nodes is used for establishing connection between autonomous systems.

ducer atomic component computes network raw data to information which is meaningful for one or more user-defined experiments. As mentioned earlier, parameters such as network throughput and average latency are observed. Measured data is stored in trace files and can be converted to graphs.

3.1.4. Databases and stacks

In the developed simulator Java implementation, a Routing Table object, which consists of a collection of Route objects, is an instance variable of the Routing Module class which in turn is an instance variable of the Node class. The Routing Table class is a vector containing references to Route objects for all routes.

Each node in the network is represented a routing table storing neighboring nodes to which traffic should be routed. Each node has a routing table for every possible destination in the network, and each table has an entry for every neighbor (see Figure 3). Data packets can be systematically routed through the network by using routing table. According to the routing algorithm, these routing tables are constructed previously (in static algorithms), dynamically adapted to network load state (in dynamic algorithms) or based on node's (insect's) next node selection probabilities to its destination - e.g., using swarm based algorithms. During simulation execution new entries may be added to table or current entries may be removed or adjusted according to network traffic. All the values of the entries in the routing table

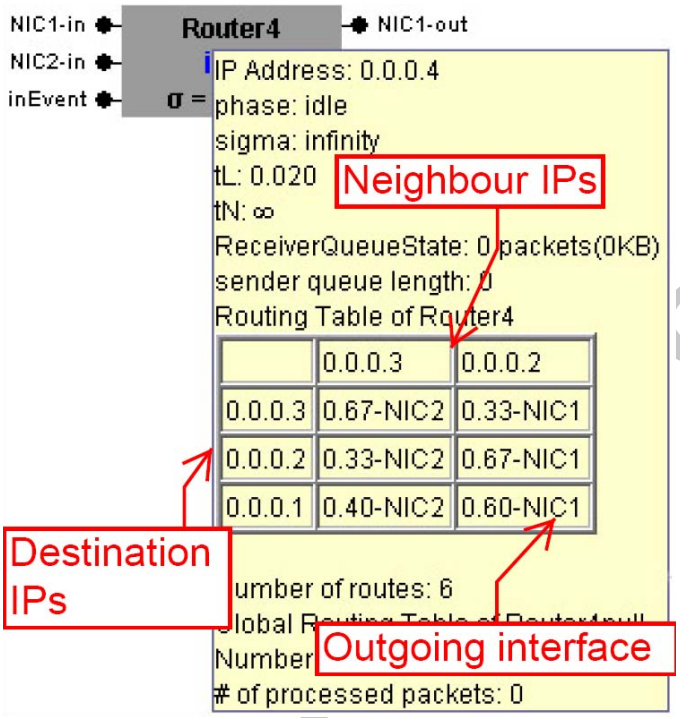


Figure 3: A Routing Table view in SwarmNet, rows corresponds to destinations and columns to neighbors

range between 0 and 1, a probabilistic value. These entries are referred to as profitability values through which most profitable routes can be chosen. This approach balances the network traffic load by routing the packets to alternative routes.

In Figure 3, the simview viewer shows the content of the routing table for the router called Router4. This is important in order to follow the formation of routing table in the execution mode for teaching network protocols logic.

As already mentioned, detailed component definitions of the SwarmNet simulator can be found in [45].

3.2. Coupled Models

Using basic components and tools which have been described above, networks can be built by coupling them under developed simulation environment (see Figure 1). Furthermore, by linking these coupled networks, larger networks can be systematically developed. Some applications have been created

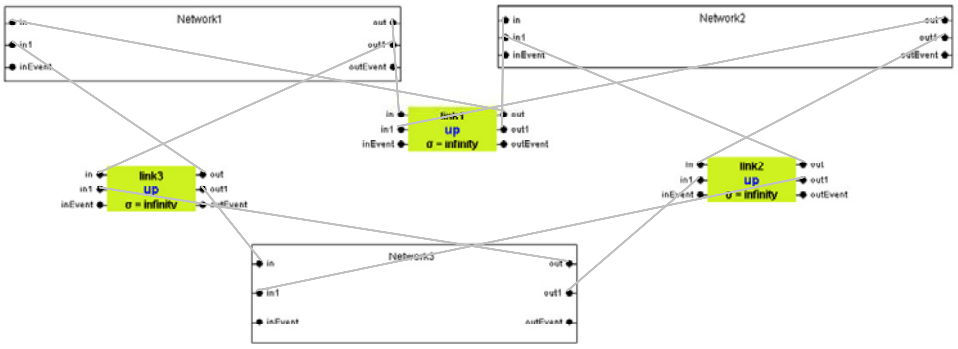


Figure 4: Three networks (a cluster) coupled together in a ring topology

in SwarmNet to simulate routing algorithms over varying network traffic patterns. In the experiments, large-scale networks with increased complexity and connectivity are built. They have been designed for testing the model design as well as testing the framework itself (e.g., scalability). Large-scale and complex networks are used for uncovering dynamics and performance measurements of the models in the SwarmNet environment. In Figure 4, three coupled models connected in ring topology. Coupled models can be considered as autonomous systems.

3.3. Clustering and Colonization

One of the main criteria for appreciating the network simulators is scalability. A network or simulator model is considered scalable with respect to network size, if simulation deserves its run properly while the number of network components such as nodes and links grows constantly. In this study, a clustering approach is employed to support scalability and implemented when coupling the models (see Figure 4). Clustering provides manageable network sizes by abstracting a subnet to a single node in a higher level network. By considering a coupled model as an atomic model, DEVS coupled model concept has a resemblance with clustering. There exists a hierarchy of networks within the total of all nodes and routers (see Figure 5). Each coupled model has a number of border nodes which are used for connecting it to other coupled networks (see Figure 2). In the approach, clustering is done in addressing level of nodes. Hierarchical and modular structure of DEVS formalism facilitates implementation of clustering approach. Border nodes have an additional routing table consisting of the cluster names. This approach substantially decreases the information stored in routers.

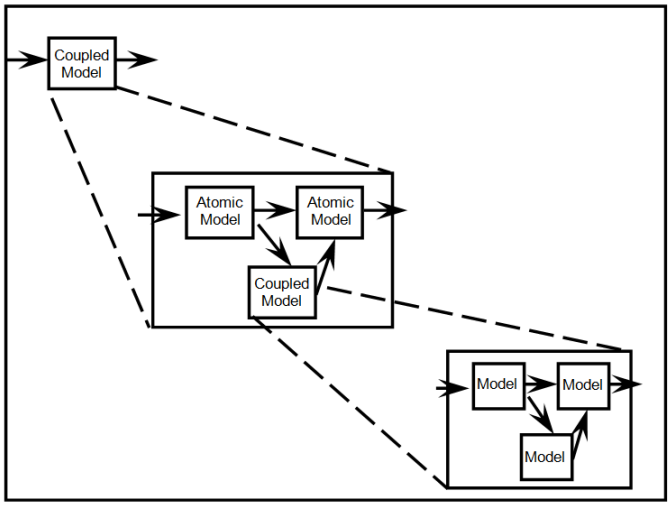


Figure 5: Modular and hierarchical design of DEVS network models

4. SwarmNet and NS-2 performance comparison

To highlight the key differences between modeling DEVS and ns-2, that relate to the goal of this paper, a sample network consisting of a small number of routers and links is used (see Figure 6). In Figure 6, the Network Animator (Nam) screenshot of the sample network is shown. Simulation experiments were performed both in ns-2 and SwarmNet.

Parameters are selected commonly for comparison purposes and listed within Table 2. Data packets in both simulation environments are configured to almost same variables, for example their sizes are set to 330 bytes and generation frequency are selected as same. In ns-2, packet events are generated via traffic agents, while SwarmNet uses a separate generator called event generator. All packets including control and data packets are modeled as a standard IP packet with 20 byte header and data fields. SwarmNet traffic model has an event generator and a transducer which generates data packets in order to mimic behavior of users and measures outcomes, respectively.

4.1. Structural differences of SwarmNet and ns-2

The simulation parameters for the SwarmNet and ns-2 environments have some relational differences which have a direct effect on simulation results. SwarmNet is a packet-level simulator, focused just only on routing, has higher

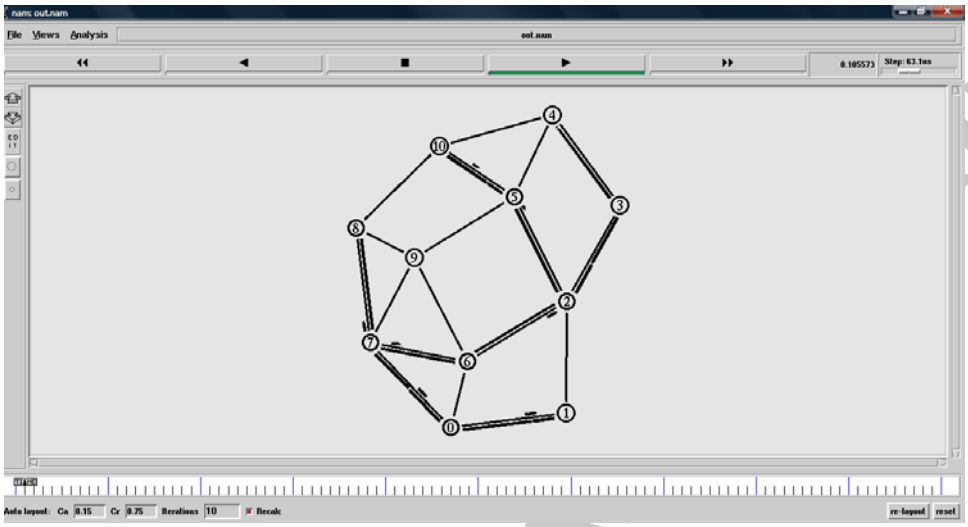
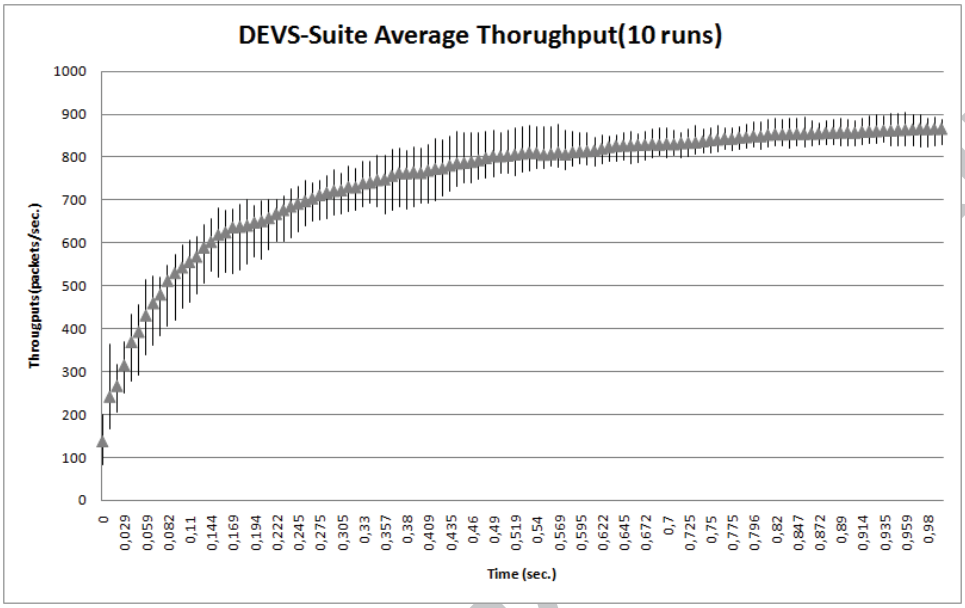


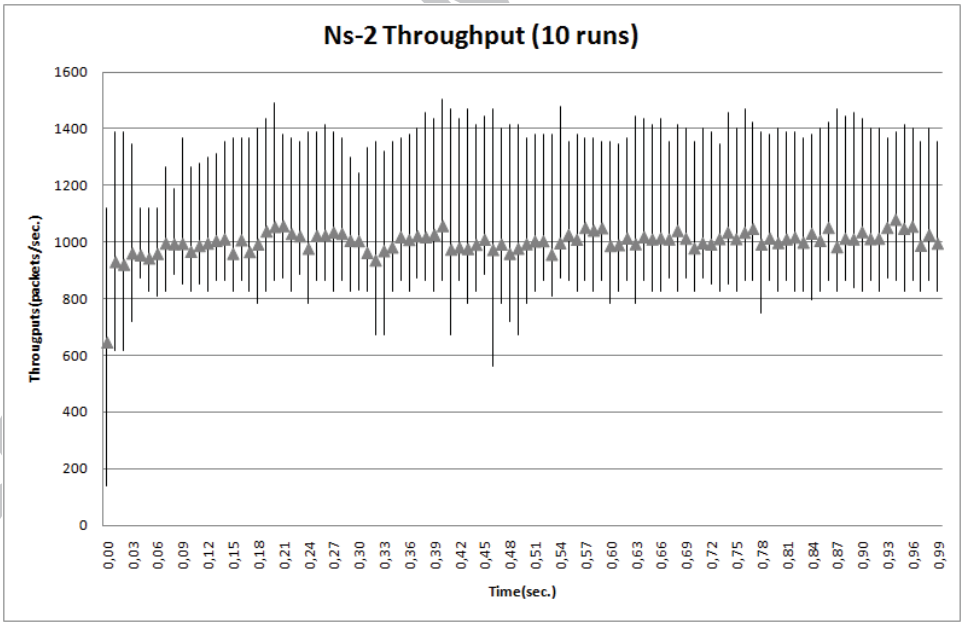
Figure 6: NAM screenshot of the simple network consisting of router nodes and duplex links

Table 2: Simulation Model Parameters of ns-2 and SwarmNet

Simulation Model Parameters		
	SwarmNet	NS-2
Topology	11 routers, 18 bidirectional links	11 routers, 18 duplex links
Protocol	Routing Information Protocol	Distance Vector
Processing speed	1 msec/event	N/A
Event frequency	1000 events/sec.	28388 events/sec
Packet sizes	330 bytes	330 bytes
Node's buffers	1 MB	Infinity
Link bandwidth	1.2 Mbps	1.2 Mbps
Link delay	2 msec.	2 msec.
Traffic type	Uniformly random	FTP over TCP
Simulation time	1 sec.	1 sec.



(a)



(b)

Figure 7: SwarmNet and ns-2 throughput comparison

level of abstraction than ns-2, finally more suitable in high performance demanding applications. In following sections, it is tried to identify the reasons behind the differences in a more comprehensive manner. Simulation results for validation purposes are shown in Figure 7 in which SwarmNet approximately yields same throughput values.

The SwarmNet and ns-2, provide different abstractions for links. In ns-2, two-way channels called full-duplex are defined such that a node can transmit and receive packets simultaneously [21]. Since simultaneous receive and send is impossible for serial processor, in SwarmNet links are defined to be bi-directional - i.e. at each point in time, a node can either send or receive data.

The ns-2 provides detailed models for state-of-the-art routing protocols such as link state and distance vector to study network protocols, based on link state and distance vector concepts, specific protocol such as OSPF and RIP can be simulated in ns-2. The RIP which corresponds to distance vector protocol is developed for the SwarmNet environment.

In SwarmNet, the router model is designed based on a basic processor with a queue. The router model incorporates a routing table and with a routing scheme that complies with the RIP protocol. Modeler selects the router model's processing time. Switching or routing delay is a time period that router takes to switch or route a packet. This time represents the total time needed for evaluating the packet header, checking the routing table which varies with table size, and assigning the packet to its intended output port of the router model. It is noted that although new hardware in IP switches can only take microseconds to route and forward packets, the router model delay time is set to 1 ms.

Unlike SwarmNet, ns-2 uses a single virtual clock to execute events that scheduled in a list. The scheduler maintains a list of events and processes ordered by time and selects next closest given the current simulation time, advances the clock to time of the event, and then executes the event to completion. The ordering of execution of events is arbitrary (see Figure 8).

Since ns-2 has supports for realistic simulation of packets and their processing, a large number of events is necessary. In SwarmNet, the router and link model behaviors are primarily determined in terms of routing algorithm not events. Consequently, after completion of the simulation, the total number of events (event frequency) in ns-2 is greater than SwarmNet by a factor of 30.

In SwarmNet, the size of the event queue is set to 1MB. With each packet

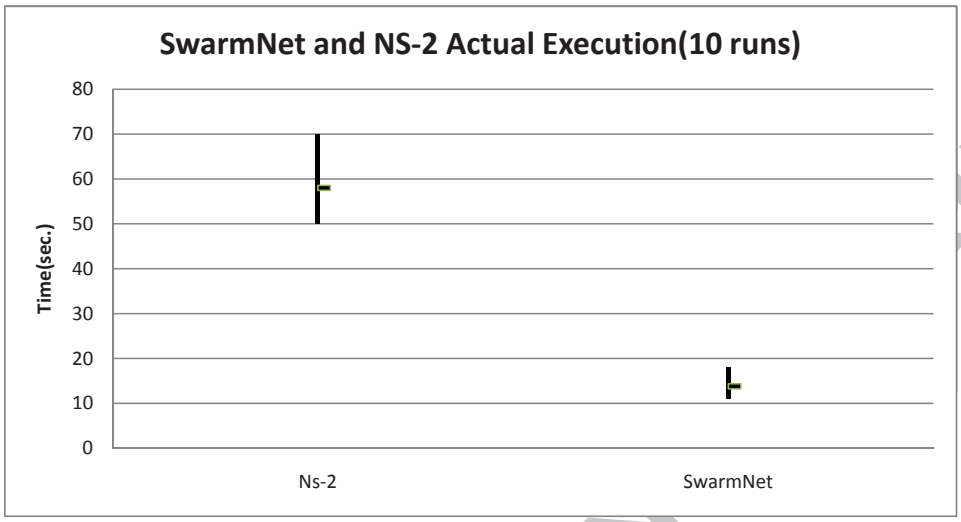


Figure 8: Wall-clock execution times of the simulators

size set to 330 bytes, approximately 3030 packets can be buffered in a node. This number of packets is more than necessary for avoiding packet losses. In the ns-2 TCP application, there is no limitation in terms of queue size. However, queue size can be set to a finite value to study network bottleneck effects.

In SwarmNet, a basic traffic model is implemented which generates data packets uniformly random sources and destinations. In ns-2, applications such as FTP which is based on TCP are supported. The traffic model behavior is comparable to that of the TCP - i.e., the traffic generated in ns-2 and SwarmNet are indistinguishable except for the hello and acknowledgement packets.

5. Large-Scale Simulation Models

To show the capability (applicability) of the biologically inspired network system modeling approach, it is started with well-known routing algorithms. For instance, static link state algorithm is modeled to initialize network and distance vectors to calculate distances between nodes and autonomous systems. Design details of these routing protocols can be found in [46] and [45].

Each simulation run consisted of an adaptation to topology phase (initialization) and a test phase under user traffic. During the initialization phase,

Table 3: Simulation Parameters of Large Models in SwarmNet

Simulation Model Parameters	
Topology	core(11 routers and 18 links) recursive (up to 4000 components)
Protocol	OSPF, RIP, BEE
Processing speed	1 msec/event
Event frequency	200 events/sec.
Packet sizes	from 1 to 100 KB random
Node's buffers	1 MB
Link bandwidth	1.2 Mbps
Link delay	2 msec.
Traffic type	Uniformly random
Simulation time	10 sec.

system runs without load and initial routing tables are formed according to the number of hops (i.e., Dijkstra [9] shortest path estimation algorithm). During the test phase, the network performance was measured and recorded in terms of average packet delay, throughput, convergence time, and packet loss ratio. In Table 3, simulation parameters are summarized. All values are chosen according to algorithm test framework in which a representative network is used to see algorithm behavior in large networks. In the following section, in order to test the algorithm across weighted conditions, all parameters are incremented.

Simulations were executed in the DEVSJAVA environment for a period of ten seconds. Using a Windows computer with 2.4GHz processor and 2GB RAM, the smallest network took a few minutes to execute whereas the largest network simulation took less than a hour to complete.

6. Performance Results

6.1. The effect of clustering on increased size networks

Network systems today are faced with increasing data and service requirements. The ability to quickly and securely access, manipulate, transfer and analyze service and maintenance of data and routing information is very important for any efficient network system. Clearly, it is impossible to store all router and link's information in a router's buffer. In other words, there is a need for design and implementation of clustering technologies because

Table 4: Simulation results of clustered versus cluster-less networks

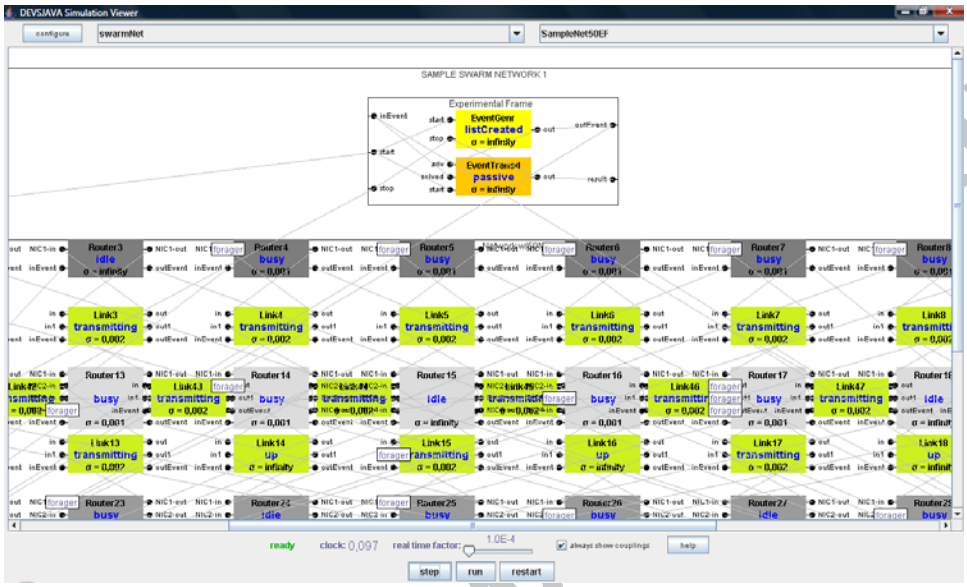
	Initialization Phase		Online Phase under Traffic				
	iteration	duration	clock	generated packets	solved packets	lost packets	average TA
cluster-less network	230	0.215	10	1999	1995	4 (% 0.2)	0.011
clustered network	101	0.087	10	1999	1999	0	0.013

large size and connectivity is far beyond of management and organization capability of current network schemes. Therefore, an approach is needing to shrink routing data and management information. This approach is called clustering which provides a fault-tolerant, scalable, and reliable network system [35]. Clustering enables network design to the modular and hierarchical and overlapping with DEVS modular approach.

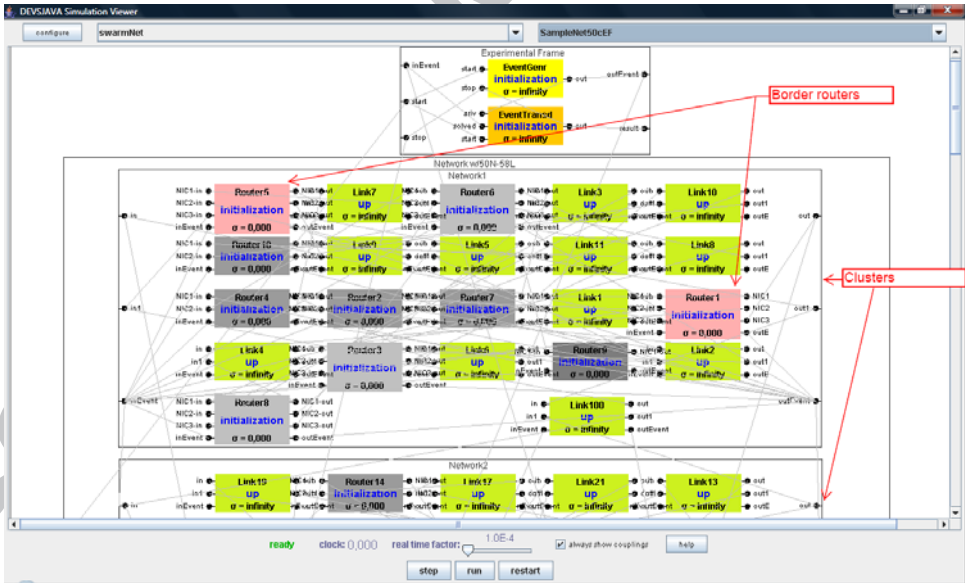
In order to show the effect of clustering on a network system, a simple experimentation is done in DEVSJAVA. First, a network with 50 routers is designed and measured its performance. Later, it is divided into 5 colonies and the results are compared. The border nodes are used in each clusters for communicating autonomous system with other networks. These two different network models are shown in Figures 9(a) and 9(b).

In Table 4, results are summarized. First effect of clustering is on initialization phase of the network which is tightly coupled with convergence of the routing protocol. As depicted in the Table 4, iteration number and convergence time of the clustered network are twice less than network without clustering. The reason is that clustering brings a kind of parallelism on network execution, and thus total process time is decreasing. Remaining values in Table 4 belong to network run under traffic. There is no lost packet in the clustering, while cluster-less case has some lost packets. This shows that clustering is more fault-tolerant as mentioned before. Finally, average turnaround values remain almost same but clustered value little bigger. In clustered network, packets are routed to destinations one step more, i.e. packets first are routed to autonomous systems in which destination router is included, therefore this decision creates little delay.

In Figure 10, throughput curves of the networks can be seen. Clustering yields two times better throughput than networks without clusters. Clustering also provides a manageable networks in which just border nodes have information about autonomous systems. In other words, routing tables of routers in an autonomous system includes entries of their neighbor routers not all routers in the whole network. This approach gradually shrinks the routing information stored in routers.



(a) 50 routers coupled network model without clustering and its experimental frame



(b) 50 routers and 5 clusters network model

Figure 9: Network model clustering

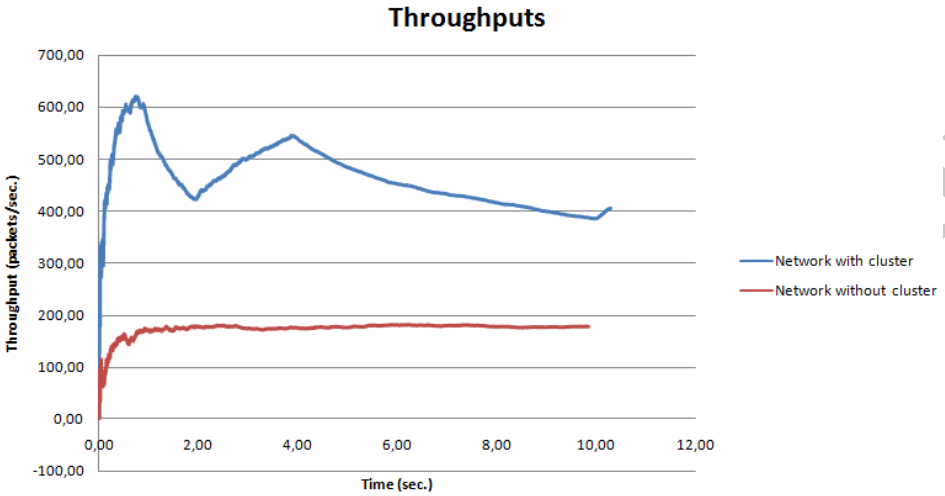


Figure 10: Comparison of the throughput values from clustered and cluster-less networks

6.2. Simulation results of large-scale network models

A primary benefit of a network-based modeling approach is its degree of support for large-scale model development and efficient simulation. Due to both scale and complexity of current network systems such as the Internet, modeling and simulation of these systems is non-trivial[13]. While scalability issue is due to the routing databases of the nodes increasing with the network size, which can cause some routers' databases to exceed their capacities, complexity comes from variety of communication media, communications equipment, protocols, and hardware and software platforms found in the network. To allow simple redesign of the routing database for large-scale networks, the above clustering approach was developed.

In order to study the scalability of the proposed approach, models for networks ranging from small networks in 28 routers to large ones in 3520 routers and links in that number are developed. Small networks were created manually while large networks were produced using a recursive topology-generating algorithm. To verify and validate the approach on larger models, a set of experiments are carried out and the results have been evaluated in a comparative manner. In these experiments, the key independent variables are the degree of network connectivity and the number of network components. Networks were modeled and their simulation results were analyzed. The relation between the network throughput and the number of nodes is

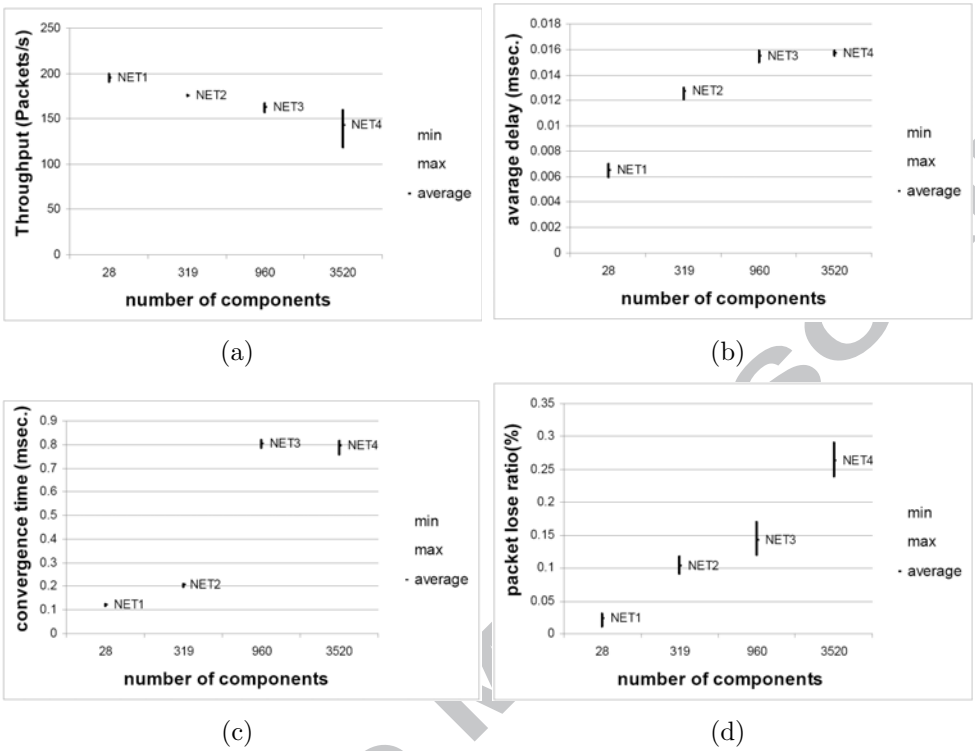


Figure 11: Large-scale network (a) throughput, (b) average delay, (c) convergence time, and (d) packet loss measurements using the BEE algorithm.

shown in Figure 11(a). The throughput gradually decreases as the number of components increases since the packet loss ratio increases in accordance with the size of the models. However, performance losses for large networks remain acceptable. Another observation is that for all network sizes, the average delay across the networks is increasing but not asymptotically (see Figure 11(b)). In implementation, the average delay or turnaround time is defined as a packet's life-span time which starts from the packet generator and ends at the packet transducer while going through the nodes and links of the network mode.

Rapid convergence is a main feature of any efficient routing algorithm. A routing algorithm must show how quickly it can construct and update the nodes' routing tables given different scales of networks. Figure 11(c) shows that the convergence time of the biologically inspired routing algorithm is scalable. Although larger networks exhibit a relatively long time to converge

as compared with smaller sized networks, their convergence times are in milliseconds. A stationary trait can be recognized in the convergence trajectory as the scale of the network approaches a thousand components. The reason is attributed to the network being composed of similar networks since larger models are recursively and automatically constructed. This approach together with the parallelism in the DEVSJAVA simulation engine causes convergence time to increase less while the number of components increases.

Finally, as shown in Figure 11(d), the packet loss ratio gradually increases with the increase in the number of components. The packet loss is shown to be linear or better as the scale of the network is increased.

7. Discussion And Evaluation

The link-state, distance-vector and swarm intelligence algorithms are used in modeling and simulation of large networks. Large network models with the BEE algorithm reach the steady state throughput, the throughput remains nearly constant to the end of the simulation. Therefore, the load balancing provided by the BEE approach is reached rapidly and evenly in the presence of heavy network traffic conditions, as well as in increasing connectivity and scale.

The ecological approach has better load balancing in large models since the probabilistic routing used in the BEE algorithm forwards the packets to alternative routes and finally balances the load. Furthermore, the network resources and the network traffic load are better utilized and evenly distributed across nodes and links even if they are large-scale. This reduces network congestions and results in the packets reaching their destinations faster. The simulation experiments show improved precision, stabilization and consistency of the hierarchical BEE routing and clustering scheme. Also, the use of random traveling of the agents (scouts) increases the robustness of the network operation.

The developed approach supports modeling and simulating adaptive, robust, and survivable network applications. Since the DEVSJAVA environment is developed using the DEVS formalism and it supports modeling of networks using the biologically derived rules instead of complex formulas, simulation models can be developed systematically and simulated efficiently. Given that the need to better understand the Internet characteristics in terms of its topology, alternative configurations, and unpredictability of

network traffic, researchers continue to develop greater capabilities to simulate large-scale models [14] and [13]. For example, simulations having more than 100,000 routers and nodes have been developed using dozens of parallel processors [27], [41] and [8].

7.1. The Key Features of the SwarmNet Network Simulator

Following treatment summarizes key features of the SwarmNet network mode which is packet-level discrete event network simulator based on DEVS Formalism. It utilizes DEVS formalism for describing network components and inherits DEVS hierarchical and modular design concepts.

7.1.1. DEVS Simulation Engine

One of the main advantages of the DEVSJAVA is its discrete event modeling infrastructure. Since the dynamics of network systems can be characterized in terms of components that can process and generate events, discrete event modeling become most appropriate approach for modeling the dynamics of network systems. Among discrete event modeling approaches, the Discrete Event Systems Specification (DEVS) 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. DEVS is suitable approach for modeling dynamic systems with intelligent parts such as biologically-inspired computer networks.

7.1.2. Network Modeling Approach

Level of abstraction have to be implemented in any modeling effort due to impossibility of bringing all parts of the nature to the computer system. It is selected to model just nodes and links as atomic models due to facts that just their states are in focus and increasing the number of atomic models and states can decrease the performance. A high level abstraction is used in which routing behavior of the network is concerned. Therefore packets such as acknowledgement and hello are ignored.

DEVS makes modeling effort systematic so that complex behavior is formed by coupling simple structured primitive models (e.g. atomic model). In other words, behavior of an atomic model does not exhibit a high level intelligence; nevertheless coupled model shows fascinating emergent behavior. Atomic models in a compact model interact via messages to form complex collective behavior.

7.1.3. Modular and Hierarchical Design

A coupled model specifies constructs for composing modular models into hierarchical structures. Behavior of a coupled model is defined by its constituent atomic (and/or coupled) models. With closure under coupling feature of DEVS, coupled models can be used as atomic models in a larger model. Coupled models can be constructed systematically using the concepts of ports and couplings between them. When a component sends messages via its output ports, the couplings relay the messages to their designated input ports. Upon receipt of messages by atomic models, they immediately process these messages which may result in new states and generation of outputs. Hierarchical and modular nature of the DEVS facilitates the implementation of both modeling the network and cluster scheme.

7.1.4. Separated, manageable, robust and accurate analysis and testing framework

To define simulation objectives, the concept of experimental frame is utilized to define the conditions under which a model can be experimented with and observed. Topologies of network models or Internet core networks can be evaluated in terms of their critical network components and their dependencies. Defined experimental frame is used for testing the model under various conditions and observing its behavior. Having been equipped with an experimental frame, simulation is run under specific experimental conditions and results are observed. The results are then evaluated in terms of whether or not they are within an acceptable range; otherwise, the model parameters are changed and simulation experiments are repeated. All statistical data from all models are collected to transducer model, therefore evaluation of the results can be done easily. Separated design of the experimental frame facilitates the tracing and analyzing the simulation results.

7.1.5. Bio-inspired robust, adaptive, scalable and collaborative design

A biologically inspired swarm routing approach is derived based on honeybees and their interactions during foraging called honeybee scout-recruit mechanism. In this framework, the movement of specialized packets such as artificial bees (called scouts) can be used to balance network loads. Given the similarity of this and agent-based approaches, each node is capable of accommodating an ensemble of scouts for controlling congestion in a distributed environment. In implementation, analogous to honeybee scout-recruit system, each network node is considered as a beehive so that bees leave their

hives for gathering nectar. Computer networks correspond to the colonies of honeybees whose goal is to find paths to profitable nectar sources. The network links correspond to the scout-recruit system where honeybees leave their hives to gather food.

Since insect societies such as bees are complex, flexible, adaptive, robust, scalable, decentralized and self-organized, approaches inspired from these societies inherit and exploit these properties. All of them are desirable features for any engineering solution.

7.1.6. Cluster-based hierarchical routing

One of the main criteria for appreciating the network simulators is scalability. A network or simulator model is considered scalable with respect to network size, if simulation deserves its run properly while the number of network components such as nodes and links grow constantly. Because Internet should be designed in a hierarchical manner for a better management, hierarchy is needed for scalability. In SwarmNet, a cluster-based hierarchical routing is invented for making memory usage lesser of simulations over very large topologies. A network topology is composed of several layers in a hierarchical manner, thus shrinking routing table size. To be able to make use of hierarchical routing for the simulations, there is a need for defining hierarchical topology and hierarchical addressing. In this study, a clustering approach is employed to support scalability and implemented when coupling the models. Clustering provides manageable network sizes by abstracting a subnet to a single node in a higher level network.

7.1.7. Visualization

Visualization support in DEVSJAVA includes tracking the creation and update of neighbor tables, routing databases and finally routing table. Although tools are provided to track the changes on routing tables, this level of visualization is not supported for example by ns-2 and difficult to code its open source design with duality of C++ and Tcl. It is possible to track and view the logic behind the routing protocol as well as discrete event-based run of the network system in DEVSJAVA environment, and therefore teaching and training of the network protocols in demanding level is done.

7.1.8. Suitability for distance education

Distance education, or distance learning, is some sort of education that focuses on technology, and web-based instructional systems design that aim to

deliver education to geographically distributed students. Distance education is an emerging field in the last decades in the world as parallel with growing network technologies. Since DEVSJAVA environment is purely Java based software, it can be employed for distance learning applications. Distance learning of network systems education can be supported via Web Start[28] application, online version of the DEVSJAVA[1] by exploiting Java Web Start technology. By this fact, students can get access to simulation environment from distant locations. DEVSJAVA yields great advantages to traditional on-campus network systems and protocol courses and provides a key functionality to courses offered in a distance-education.

Furthermore, distributed capability of DEVSJAVA makes support for the concepts like interoperability, model repository and reusability which are not supported in most of the frameworks.

7.1.9. Reusability

Code reuse of models and components is a very good way to increase model development productivity and application maintainability. Instead of developing the all things, one should always look for well-designed models, application frameworks and customizable components. Reusability of developed models in DEVSJAVA environment allows modeler to make rapid changes, exploit new features globally, and spend less time testing and debugging in course practices.

8. Conclusions And Future Marks

In this paper, biologically inspired modeling constructs are incorporated into the general-purpose DEVS modeling framework for large-scale experiments. The resulting DEVSJAVA modeling approach affords scalable and efficient simulation of computer network systems. The proposed approach shows better response time for discovery and deployment of new routes and affords higher robustness. The use of the control (scouts) packets does not play a significant role in the total network load due to their lightweight design, therefore it is possible to manage large amount of routers and links. In the case of network malfunction such as link unavailability or node congestion (i.e., node has reached the maximum number of packets it can process), the network remains functional since the probabilistic routing adapts faster to fluctuations in the network and can find alternative paths for destinations at

run-time. Based on these observations, the network has higher survivability against surges.

From the perspective of model specification, the node and link models can be extended to use probabilistic timing and include security features. From the application vantage point, it would be interesting to apply this approach to modeling crowd behaviors since existing simulation environments lack the underlying formal theory provided by DEVS. Finally, this DEVSJAVA modeling approach can support design of emergent and scalable network systems and can be simulated in distributed and/or service-oriented computing technologies.

References

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

[2] C. Anderson. The adaptive value of inactive foragers and the scout-recruit system in honey bee apis mellifera colonies. *Behavioral Ecology*, 12(1):111–119, 2001.

[3] S. Appleby and S. Steward. Mobile software agents for control in telecommunications networks. *BT Technology Journal*, 12(2), 1994.

[4] G. Beni and J. Wang. Swarm intelligence in cellular robotic systems. In *Proceeding of NATO Advanced Workshop on Robots and Biological Systems*, pages 122 –129, Tuscany, Italy, June 2630 1989.

[5] E. Bonabeau, M. Dorigo, and G. Thraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, 1999.

[6] A. Boukerche, Z. Ming, and A. Shadid. DEVS Approach to Real-time RTI Design for Large-scale Distributed Simulation Systems. *Simulation-Transactions of The Society For Modeling And Simulation International*, 84(5):231–238, 2008.

[7] K.-Y. Cai and B.-B. Yin. Software execution processes as an evolving complex network. *Information Sciences*, 179(12):1903 – 1928, 2009. Special Section: Web Search.

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

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

[10] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics Part B*, 26(1):1–13, 1996.

[11] M. Dorigo and T. Sttzle. *Ant Colony Optimization*. MIT Press, 2004. ISBN 0-262-04219-3.

[12] 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.

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

[14] R. Fujimoto, K. Perumalla, A. Park, H. Wu, M. Ammar, and G. Riley. Large-scale network simulation: how big? how fast? In *MASCOTS 2003 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems*, pages 116–123, 2003.

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

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

[17] M. Heusse, D. Snyers, S. Gurin, and P. Kuntz. Adaptive agent-driven routing and load balancing in communication networks. In *Advances in Complex Systems*, pages 15–16, 1998.

[18] I. Kassabalidis, M. El-Sharkawi, I. Marks, R.J., P. Arabshahi, and A. Gray. Swarm intelligence for routing in communication networks. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, volume 6, pages 3613–3617 vol.6, 2001.

[19] 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.

[20] F. Lin, H. Ying, R. MacArthur, J. Cohn, D. Barth-Jones, and L. Crane. Decision making in fuzzy discrete event systems. *Information Sciences*, 177(18):3749 – 3763, 2007.

[21] ns 2. The ns-2 network simulator. http://www.isis.edu/nsnam/ns/, 2010.

[22] ns 3. The ns-3 network simulator. http://www.nsnam.org/, 2010.

[23] OPNET. OPNET simulator. http://www.opnet.com/, 2010.

[24] K. Parsopoulos and M. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2-3):235–306, 2002.

[25] D. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi. The bees algorithm. *Technical Note*, UK, 2005.

[26] Ptolemy. Ptolemy project. http://ptolemy.eecs.berkeley.edu/ptolemyII/, 2010.

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

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

[29] 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.

[30] H. Sarjoughian and B. Zeigler. DEVSJAVA: Basis for a DEVS-based collaborative M&S environment. In *SCS Western Multi-Conference*, volume 5, pages 29–36, San Diego, CA, 1998.

[31] H. S. 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.

[32] R. Schoonderwoerd. Collective intelligence for network control. Master's thesis, Delft University of Technology, Faculty of Technical Informatics, 1996.

[33] T. Seely. *The Wisdom of the Hive*. Harvard University Press, Cambridge, 1995.

[34] Y. Shi, H. Liu, L. Gao, and G. Zhang. Cellular particle swarm optimization. *Information Sciences*, In Press, Corrected Proof:–, 2010.

[35] M. Steenstrup. *Routing in Communications Network*. Prentice-Hall, 1995.

[36] C. Tovey. The honey bee algorithm: A biological inspired approach to internet server optimization. *Engineering Enterprise, the Alumni Magazine for ISyE at Georgia Institute of Technology*, pages 13–15, Spring 2004.

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

[38] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, and Q. Tian. Self-adaptive learning based particle swarm optimization. *Information Sciences*, In Press, Corrected Proof:–, 2010.

[39] T. White, B. Pagurek, and F. Oppacher. Connection management using adaptive agents. In *International Conf. on Parallel & Distributed Processing Techniques & Applications*, pages 802–809, 1998.

[40] Wikipedia. Swarm intelligence. http://en.wikipedia.org/wiki/Swarm_intelligence, 2009.

[41] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *IEEE Infocom*, volume 2, pages 594–602, San Francisco, CA, March 1996. IEEE. URL `http://www.cc.gatech.edu/fac/Ellen.Zegura/papers/howto.ps.gz`.

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

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

[44] A. Zengin, H. Sarjoughian, and H. Ekiz. Study of biologically-inspired network systems: Mapping colonies to large-scale networks. In *European Modeling & Simulation Symposium (EMSS)*, pages 537–545, Campora S. Giovanni, Italy, September 17 - 19 2008.

[45] A. Zengin, H. Sarjoughian, and H. Ekiz. Discrete event modeling of swarm intelligence based routing in network systems. A Revision Submitted to Information Sciences Journal, 2010.

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