STUDY OF BIOLOGICALLY-INSPIRED NETWORK SYSTEMS: MAPPING COLONIES TO LARGE-SCALE NETWORKS

Ahmet Zengin[(a)], Hessam Sarjoughian[(b)], Hüseyin Ekiz[(c)]

[(a), (c)] Sakarya University Technical Education Faculty
Department of Computer Science Education
Serdivan, Sakarya, TURKEY
[(b)]Arizona State University
School of Computing & Informatics
Arizona Center for Integrative Modeling and Simulation
Tempe, AZ, USA

[(a)]azengin@sakarya.edu.tr , [(b)]sarjoughian@asu.edu, [(c)]ekiz@sakarya.edu.tr

ABSTRACT
Natural systems can offer important concepts for modeling network systems. A biologically-inspired discrete-event modeling approach is described for studying networks' scalability and performance traits. 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. We describe our approach which is based on mapping the behavior of the honeybees to discrete event models. Large-scale network models are simulated and evaluated to show the benefits of nature-inspired network models.

Keywords: beehive, DEVS, networks, routing, scalability, simulation.

1. INTRODUCTION

Network systems must communicate with one another using a variety of algorithms and technologies. Many systems supporting interconnectivity are required to exhibit essential traits such as adaptability, scalability, and reliability (survivability). At the same time, these networked systems are expected to offer new and more sophisticated services in the face of increasing system heterogeneity (Lunceford and Page 2002). To cope with the management of such networks in the presence of ever increasing complexity, various decentralized and centralized approaches are being used to address the needs of private and public organizations (Steenstrup 1995).

Ecological models are being studied for development and operation of decentralized (distributed) systems such as communication networks. Large-scale biological systems, such as bee colonies, have advanced mechanisms that are scalable and adaptable under varying environmental conditions (Bonabeau, Dorigo, and Theraulaz 1999). 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 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 and mechanisms to the design and implementation of 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 and compared to the current state-of-the-art distance vector algorithm have been shown to result in the highest throughput and lowest delays in Internet-style networks. We have devised a new class of agent-based routing algorithm based on principles of biological swarms, which have the potential to address some of the above problems in an autonomous and intelligent fashion.

To develop and study dynamic and adaptive swarm-based routing protocols, we have devised a DEVS (Discrete Event System Specification) (Zeigler, Praehofer, and Kim 2000) network model called SwarmNet. Discrete event simulation is a resilient, powerful and efficient computational device that support exploring the behaviors of complex systems. These models and networks are implemented in the DEVSJAVA modeling and simulation environment (Zeigler and Sarjoughian 2003, ACIMS 2008) which is an implementation of the DEVS framework.

The nodes and links are characterized as the elementary network components. Using the DEVS hierarchical model composition concept, we develop simulation models of networks with varying topologies and scales. For example, we will use clusters to study its impact on reducing communication and increasing

performance. The explicit and hidden behaviors of these networks are observed under various experimental configurations – e.g., nodes and links are assigned different capacities. Lastly, we also examine the performance of the DEVSJAVA environment for networks having from tens to several thousands of components and connections.

In the remainder of this paper, starting in Section 2, we briefly review some related key features of SwarmNet and highlight its strengths and weaknesses. In Section 3, we describe the modeling concepts from honeybees are mapped to a set of adaptable agent-based DEVS modeling constructs. In Section 4, we develop and analyze example models in the SwarmNet simulation environment. In Section 5, we summarize our work and present some future research directions.

2. KEY FEATURES OF THE SWARMNET

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

DEVS Simulation Engine: 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) (Zeigler, Praehofer, and Kim 2000) 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 (X, S, Y, δext, δint, δconf, λ, ta). The external (δext), internal (δint), confluent (δconf), output (λ), and time advance functions (ta) define a component's behavior over time (for examples of the atomic model see the Listings 1 and 2). 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 DEVSJAVA (ACIMS 2008). DEVSJAVA is an object-oriented realization of Parallel DEVS. It supports describing complex structures and 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 sequential, parallel, or distributed simulation engines using alternative computational environments such as CORBA, HLA, and Web-services are important considerations. Furthermore, the DEVS models are extended with other kinds of models such as fuzzy logic (Sarjoughian and Cellier 2001).

Network Modeling Approach: 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. In order for modeling a distributed networked system, we have defined a set of basic network simulation model components including nodes which communicate with one another via links. By coupling these model components in DEVSJAVA, we can develop a variety of network configurations and study network characteristics. Since it is assumed that only nodes and links of a network are able to cause bottleneck, they are modeled as atomic models and only their states as well as input and output variables are of interest. Other network components such as packets and routing tables are realized and modeled as stateless entities.

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 (Wymore 1993). Upon receipt of messages by atomic models, they immediately process these messages which may result in new states and generation of outputs.

Robust and accurate analysis and testing framework: To define simulation objectives, we utilize the concept of experimental frame an experimental frame 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. In this work, a typical experimental frame consists of generator and transducer atomic models. We employ generator in order to create network traffic and to schedule special events such as

unavailability of links or nodes. To realize this, a generator model sends messages to all the appropriate network components. 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 in under specific experimental conditions and results are observed. The results are then evaluated in terms of whether or not they are within in acceptable range; otherwise, the model parameters are changed and simulation experiments are repeated.

Bio-inspired robust, adaptive, scalable and collaborative design: As stated above, a swarm routing approach which is biologically inspired model is derived based on honeybees and their interactions called honeybee scout-recruit mechanism during foraging. 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 our implementation, analogous to honeybee scout-recruit system, each network node is considered as a beehive so that bees leave their hives for gathering nectar. Network corresponds to the world of honeybees who seek richer nectar sources, finding paths with higher capacities to result in nectar sources that have higher profitability. Control packets in the form of light-weight scout entities searching for nectar and foraging for information to aid survival of the network (honeybee colonies). Each hive deploys a number of scouts to find the most profitable paths for a given destination. Each router then uses the information received from all the nodes in the network obtained by its scouts to calculate the shortest path to each destination in terms of a chosen metric. Scouts control congestion by making alterations to routing tables in order to route new traffic away from congested nodes. Then, packets are dispatched from a source to a destination according to information gathered by scouts.

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 grows constantly. Because Internet should be designed in a hierarchical manner for to be better managed, hierarchy is needed for scalability (Zeigler and Mittal 2002). Cluster-based hierarchical routing was 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 need for defining hierarchical topology and hierarchical addressing. In this study, we employed a clustering approach to support scalability and implemented when coupling the models. Clustering provides manageable network sizes by abstracting a subnet to 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. Each coupled model has a number of border nodes which are used for connecting it to other coupled networks. In our 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.

3. SwarmNET FRAMEWORK

We have defined a set of basic network simulation model components including nodes which communicate with one another via links (see Figures 1) as detailed next. By coupling these model components in DEVSJAVA, we can develop a variety of network configurations and study their characteristics. Since it is assumed that only nodes and links of a network are able to cause bottleneck, they are modeled as atomic models and only their states as well as input and output variables are of interest. Other network components such as packets and routing tables are realized and modeled as stateless entities.

3.1. Network Model Specification

The nodes in the network are modeled as a DEVS atomic component. Each node has several inputs and outputs through which messages among nodes can be received and sent (see Figure 1). 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. The main part of the nodal structure is the network interface (NIC) that provides fundamental inter-networking services.

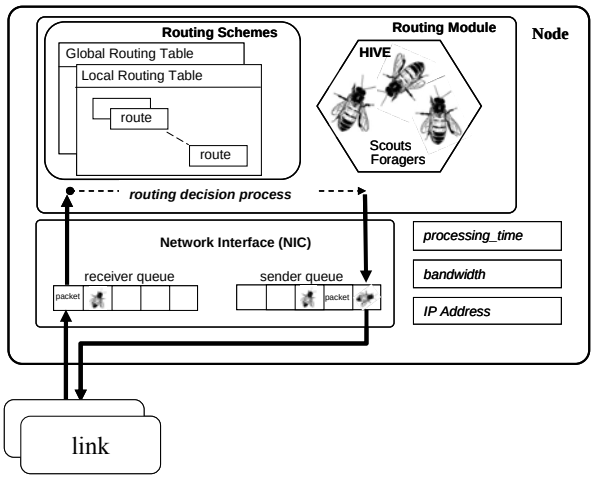


Figure 1: Node Model

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. Routing Module includes a routing table for local network as well as a global routing table which can be used to manage the routing between the local network and other parts of the global network. In our swarm application, beehive is configured to launch scouts, foragers, and other bees to monitor and reconfigure network resources. All link models are capable of accommodating different kind of entities for supporting different network designs.

The link is modeled as an atomic model. It has a central role in defining networks having different topologies. All links are communication channels and therefore are viewed as bit-pipes which are characterized with bandwidth (bits/sec) and transmission or propagation delay specified in milliseconds (see Figure 2). Each link is defined to be bidirectional and thus supports concurrent bidirectional interactions. Each duplex link has some finite capacity. The packets that arrive are placed in the queues and are transmitted to the next node using first-in first-out (FIFO) strategy. Links are able to carry traffic of a certain bandwidth up to the total capacity of the link. The specification of the link is akin to a simple processor with a queue that can process incoming packets according to FIFO or some other discipline. Each link has input and output ports for connecting two nodes in a duplex manner (see for example Link 1 atomic model in Figure 2).

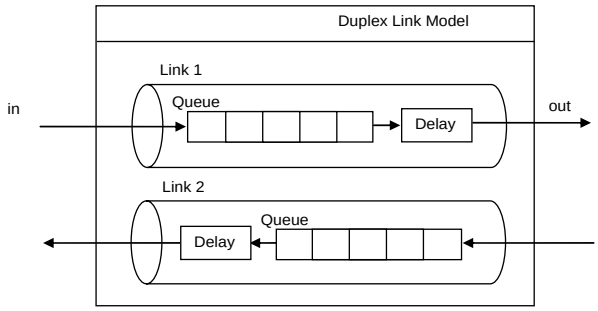


Figure 2: Bidirectional Link Model

Other network components such as routing tables and packets modeled in DEVSJAVA as stateless models. In Figure 3, the DEVSJAVA viewer shows the content of the routing table for the router called Router4. This is important both following the formation of routing table in the execution mode. Packets are modeled as a DEVS messages and categorized as control and data packets. Control packets can carry bee agents.

3.2. Honeybees and Network Conceptual Models of Beehive

In this work, a swarm routing approach derived from honeybees and their interactions was developed using the concept of the honeybee scout-recruit mechanism during foraging. The starting point for developing the biologically inspired approach for modeling and simulating network systems is that "real bees can find optimum solutions in their foraging activity" (Seely 1995). Table 1 shows the analogy between honeybee colonies and computer networks.

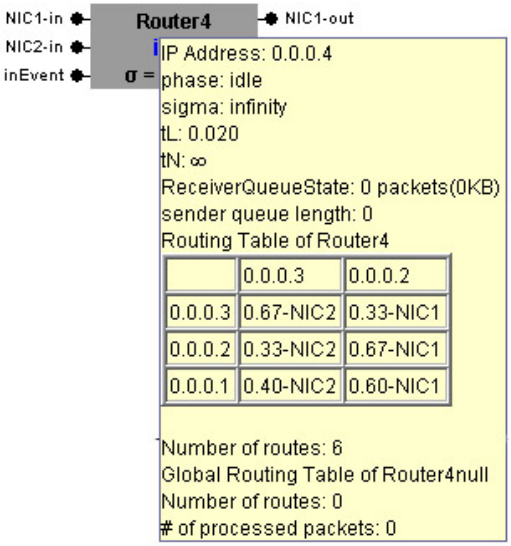


Figure 3: A Routing Table view in DEVJAVA, rows corresponds to destinations and columns to neighbors.

Computer networks correspond to the colonies of honeybees whose goal is to find paths to profitable nectar sources. Each network node is analogous to a beehive. The network links correspond to the scout-recruit system where honeybees leave their hives to gather food. Network nodes exchange control and data packets. Data packets correspond to foragers carrying nectar to beehives. The router inside each node uses the information received from all the nodes in the network (obtained by its control packets) to calculate the shortest path to each destination in terms of a chosen metric. The network links are used to limit the number of control and data packets that can be sent and received between nodes. This abstraction corresponds to the amount of information scouts and foragers can carry while searching for and transporting nectar. Each network node deploys a number of control packets (scouts) to find the most profitable paths for a given destination.

Table 1: Analogy between Network Systems and Honeybee Colonies

Computer Networks	Honeybee Colonies
nodes	beehive
network	nectar collecting area
spare network resources	nectar
links	flying to/from nectar/hive
control packets	scouts
data packets	foragers
routing information	dances and cues
Inter-network control packets	drones

The BEE (scout-recruit) routing algorithm is defined by the following rules:

Rule 1: Periodically or in event-triggered way, each node dispatches scout bees for gathering information about network for choosing the best route for sending packets over the network as in scouting the nectar during foraging.
Rule 2: During foraging, the goal of each scout is to collect nectar as much as possible.
Rule 3: Scouts then wonder around the network and gather information about the status of the network.
Rule 4: Given each node and its routing table, scouts choose neighbor nodes with estimated probabilities.
Rule 5: In the network, the nectar quantity can be collected by walking along a certain route is inversely proportional to the route cost.
Rule 6: Scouts are delayed at congested links.
Rule 7: Once a scout has reached its destination, it goes back to its source. During going back, it has high queuing precedence by which delay on links is kept less.
Rule 8: A scout never visits a node twice. To do so, scouts save a tabu list which includes the IPs of the visited node.
Rule 9: A scout never uses a link that does not have enough bandwidth.
Rule 10: A scout dies after it had reached its maximum number of hops.
Rule 11: When all or some scouts have returned to their beehives, the costs of found or recorded paths are evaluated and entered in routing table.

Self-organization of artificial bees is based on these relatively simple rules for derived from individual insect's behavior. These artificial bees correspond to a special class of automata called scout-recruit system that react to their local perception of the environment by stochastically adopting predefined behaviors. Autonomous actions are committee depending on local information and local interactions.

Control packets help control congestion by making alterations to routing tables in order to route new traffic away from congested nodes. Control packets also play a crucial role in ensuring survivability of the network based on the same principle honeybee colonies maintain their existence – i.e., by using scouts to search for nectar. Furthermore, the movement of control packets is used to balance network loads. Therefore, each node is capable of accommodating an ensemble of scouts for controlling network congestion.

4. SIMULATION MODELS

To show the capability (applicability) of the biologically inspired network system modeling approach, we started with well-known routing algorithms which are also used implemented. For instance, static link state algorithm is modeled to initialize network and distance vectors to calculate distances between nodes. In the implementation of all these algorithms, we used hop count as a metric although other metrics such as available link bandwidth may also be used.

Since one of the main objectives of this approach was to test performance of the new routing approach against the state-of-the-art algorithms, a representative network was used (see Figure 4). As we will show in the other example applications, by changing the number of node and link models and varying their parameters, alternative network topologies can be readily created. The network model used for comparing the above algorithms has 11 nodes (i.e., n1, ... , n11) and 18 bidirectional links (link1, ..., link18) – this is the representative network.

Each simulation run consisted of an adaptation to topology phase (initialization) and a test phase. During the initialization phase, system runs without load and initial routing tables are formed according to the number of hops (i.e., Dijkstra shortest path estimation algorithm (Dijkstra 1959)). 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 2, simulation parameters are summarized. All values are chosen according to algorithm test framework in which a representative network is used to see algorithm behavior and compare with other routing algorithms In the following section, in order to test algorithm across weighted conditions, all parameters are incremented.

Table 2: Simulation Model Parameters

Simulation Model Parameters	
Topology	11 routers, 18 bidirectional links
Simulation time	1 sec.
Node's buffers	1 MB
Node processor speed	1 msec/event
Link bandwidths	1.5 to 6 Mbps
Link delays	1 to 5 msec.
Traffic type	Uniformly random
Event frequency	1000 event / sec.
Packet sizes	10 to 100 KB

We used two standard performance metrics: throughput and packet delay. The amount of network traffic is determined by the number of packets in the network. Generally, many packets must wait in limited capacity (FIFO queue) for processing at the nodes. We avoided generation of packets with the same source and destination nodes, although this can be done as long as there is no direct source to destination connectivity (i.e., at least one link is used between the source and destination nodes).

Simulation Results & Discussion: We compare our approach with the RIP (Routing Information Protocol) algorithm which is commonly used in today's Internet. The BEE routing algorithm is shown to yield approximately 15.94% better throughput compared with

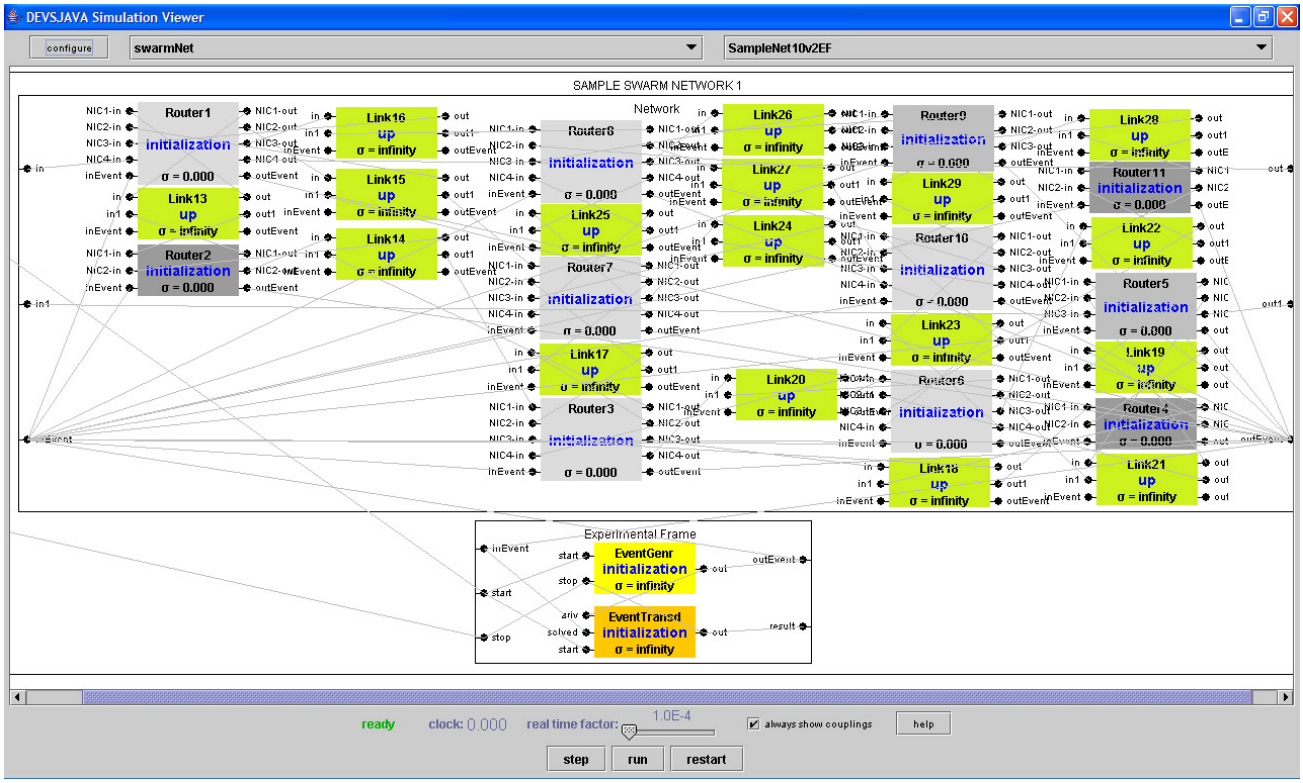


Figure 4: An Experimental Frame connected with its Network Model under DEVSJAVA

the RIP under optimum traffic load balancing (i.e., the BEE algorithm handles 916.34 packets/sec while the RIP algorithm can handle 790.5 packets/sec). Since throughput in the BEE algorithm reaches its maximum value in a shorter time (200 milliseconds), the BEE algorithm has better response time compared with the RIP algorithm. Once the network model with the BEE algorithm reaches the steady state throughput, the throughput remains nearly constant to the end of the simulation. In comparison, the network model with the RIP algorithm reaches its maximum throughput at a much later time (500 milliseconds). Therefore, the load balancing provided by the BEE algorithm is reached rapidly and evenly in the presence of heavy network traffic conditions.

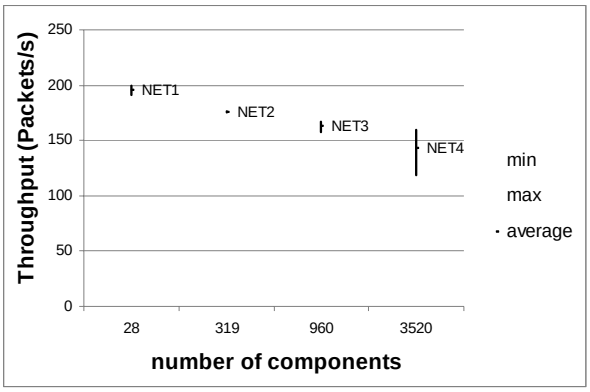
We also compared the response times for each of these algorithms based on the turnaround time of the packets that are transmitted through the network. In our implementation, the turnaround time is defined as a packet's life-span time which starts from the packet generator and ends at the packet transducer while going through the nodes and links of the network mode. Average packet delays for the BEE and RIP algorithms are 7 and 9 milliseconds, respectively. The primary reason for this difference is attributed to the scouts in the BEE algorithm. The scouts find optimum routes quickly and thus allow faster response time to network changes. Bees keep the traffic low relatively to RIP. The ecological approach has better load balancing since the probabilistic routing used in the BEE algorithm forwards the packets' alternative routes. Furthermore, the network resources are better utilized and thus the network traffic load is distributed evenly across nodes and links. This reduces network congestions and results in the packets reaching their destinations faster. The simulation experiments show improved precision, stabilization and consistency of the BEE routing 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 SwarmNet 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 (Fujimoto et al. 2003, Floyd and Paxon 2001). For example, simulations having more than 100,000 routers and nodes have been developed using dozens of parallel processors (Riley, Fujimoto, and Ammar 1999; Zegura, Calvert, and Bhattacharjee 1996; Cowie, Nicol, and Ogielski 1999).

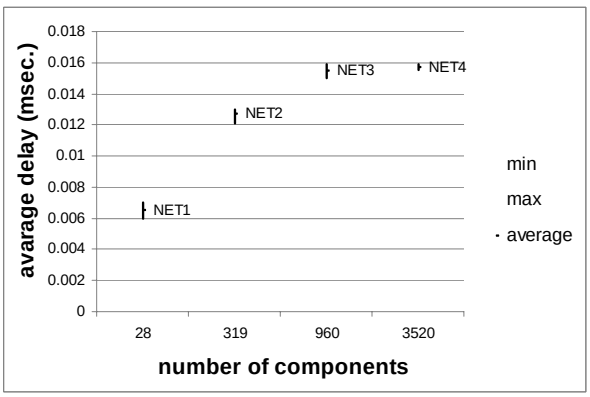
Simulation experiments of large-scale network models: A primary benefit of a network-based modeling approach is its degree of support for large-scale model development and efficient simulation. Due to both scale and complexity of current network systems such as the Internet, modeling and simulation of these systems is non-trivial (Floyd and Paxson 2001). 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.

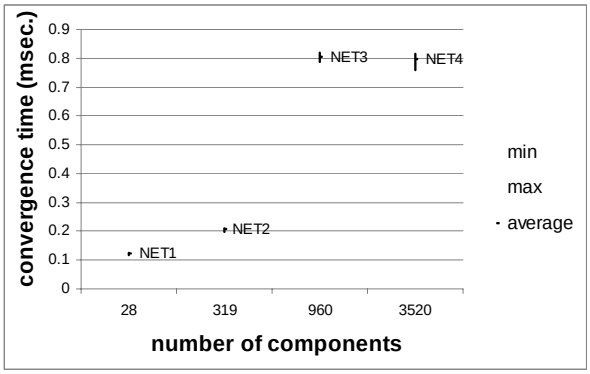
In order to study the scalability of the proposed approach, we developed models for networks ranging from 29 to 3520 components. Small networks were created manually while large networks were produced using a recursive topology-generating algorithm. To verify and validate the approach on larger models, a set of experiments were 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 shown in Figure 5(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 increasing (see Figure 5(b)).



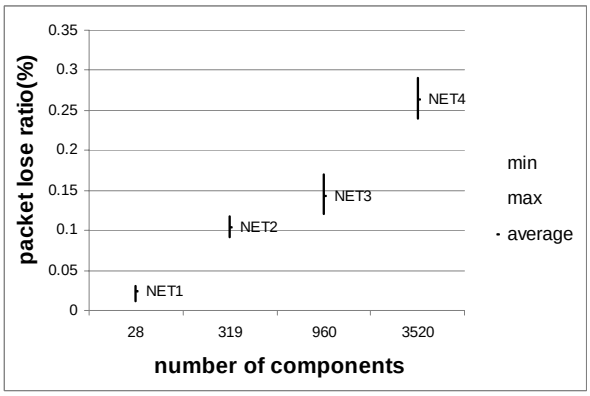
(a)



(b)



(c)



(d)

Figure 5: Large-scale network (a) throughput, (b) average delay, (c) convergence time, and (d) packet loss measurements using the BEE algorithm.

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 5(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 5(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. Simulations were executed in the DEVSJAVA environment for a period of ten seconds. Using a Windows computer with 2.4GHz processor and 512M RAM, the representative network took a few minutes to

execute whereas the largest network simulation took less than three hours to complete.

5. CONCLUSIONS

In this paper, we incorporated biologically inspired modeling constructs into the general-purpose DEVS modeling framework. The resulting SwarmNet modeling approach affords scalable and efficient simulation of computer network systems. The developed BEE routing algorithm derived from the concepts found in the social insect societies show better performance compared with the commonly used RIP algorithm. 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. 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 SwarmNet modeling approach can support design of emergent and scalable network systems and can be simulated in distributed and/or service-oriented computing technologies.

REFERENCES

ACIMS. Arizona Center for Integrative Modeling and Simulation. 2008. http://www.acims.arizona.edu/SOFTWARE/software.shtml

Bonabeau, E., Dorigo, M., and Théraulaz, G., 1999, *Swarm Intelligence: from natural to artificial systems*, Oxford University Press.

Cowie, D., Nicol, M., and Ogielski, A. T., 1999. Modeling the global internet. *Computing in Science and Engineering*, vol. 1, no. 1, pp. 42–50.

Dijkstra, E.W., 1959. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik* Vol. 1.

Floyd and Paxson, V., 2001. Difficulties in Simulating the Internet, *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 392–403.

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

Lunceford, W.H. and Page, E.H., 2002. *Grand Challenges for Modeling and Simulation*, Editors, Western Multiconference, San Antonio, TX.

Riley, G., Fujimoto, R., and Ammar, M. H., 1999. A generic framework for parallelization of network simulations, simulation: how big? how fast?. *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems(MASCOTS)*, pp. 128–, 1999.

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

Seely, T.D., 1995. *The Wisdom of the Hive*. Cambridge, Mass: Harvard University Press.

Steenstrup, M. E. (Ed.)., 1995. *Routing in Communications Network.* Prentice-Hall.

Wymore, W.A. 1993. *Model-based Systems Engineering: An Introduction to the Mathematical Theory of Discrete Systems and to the Tricotyledon Theory of System Design*, Boca Raton, CRC.

Zegura, Calvert, K. L., and Bhattacharjee, S., 1996. How to model an internetwork, *IEEE INFOCOM*, vol. 2. pp. 594–602, March 1996, San Francisco, CA

Zeigler, B.P., Praehofer, H., and Kim, T.G., 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Second Edition Academic Press.

Zeigler, B.P., Sarjoughian, H.S., 2003. *Introduction to DEVS Modeling & Simulation with JAVA: Developing Component-based Simulation Models*, Available from: http:// www.acims.arizona.edu/PUBLICATIONS.

Zeigler, B.P., Mittal, S., 2001. Modeling and Simulation of Ultra-large Networks: A Framework for New Research Directions. *ULN Workshop*, July 2002.

Zengin, A., Sarjoughian, H.S, and Ekiz, H., 2004. Biologically Inspired Discrete-Event Network Modeling. *Proceedings of the European Simulation Symposium*, pp. 317-324, Budapest, Hungary, Oct. 17-20.

AUTHORS BIOGRAPHY

AHMET ZENGIN is Assistant Professor at Sakarya University, Turkey. His experience with modeling and simulation includes a one-year-stay in ACIMS Lab at the Arizona State University. His research topics include DEVS theory, multi-formalism modeling, parallel and distributed simulation, modeling and simulation of large-scale networks, distributed systems management, biologically-inspired optimization schemes. His main research interest lies in parallel and distributed simulation and the High Level Architecture.

HESSAM S. SARJOUGHIAN is Assistant Professor of Computer Science and Engineering at Arizona State

University, Tempe and Co-Director of the Arizona Center for Integrative Modeling and Simulation. His research includes modeling theory, multi-formalism modeling, collaborative modeling, distributed co-design, intelligent agents, and software architecture. His professional experience has been with Honeywell and IBM. Visit <http://www.eas.asu.edu/~hsarjou> and <http://www.acims.arizona.edu> for more information.

HUSEYIN EKIZ is received M.Sc. in 1993 from Gazi University, Turkey, and Ph.D. degree in computer engineering in 1998 from the University of Sussex, England. He is currently Professor of the Department of Computer Systems Education and Dean of the Technical Education Faculty, Sakarya University, Turkey. His research interests are in the fields of network systems, distance education, digital circuit design and microprocessor architectures.