Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

ONE-TO-ONE MODELING AND SIMULATION OF UNBOUNDED SYSTEMS: EXPERIENCES AND LESSONS

Rohyt V.Belani
Saumitra M.Das

Information Networking Institute
Carnegie Mellon University
Pittsburgh, PA 15213,U.S.A.

David Fisher

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.

ABSTRACT

Conventional computer modeling and simulation have focused on computer objects that represent elements of the real world. In this paper we present a new approach to modeling and simulation in which authors describe the characteristics of the world being simulated without specifying how they are to be represented as computer objects. This approach is enabled by the EASEL modeling and simulation system (EMSS). Furthermore, this approach does not assume global visibility and centralized control, which are inherently inaccurate assumptions for unbounded systems (in which participants have incomplete or imprecise information about the system as a whole). Because this approach allows models to be one-to-one with the real world, models should be more accurate and simulations more realistic. The discussion includes the challenges faced in the modeling and simulation process of the distance vector IP routing protocol over a large-scale communications network, and the language features intended to address these problems.

1 INTRODUCTION

The process of simulation is used to yield realistic results for analysis of a system or protocol before its actual deployment in the real world. The success of the simulation process relies heavily on the accuracy of the underlying model of the system, which can be measured by its proximity to reality. Thus accuracy of models is imperative for successful design and implementation of the real world system being simulated. The accuracy of the model is of special importance for systems that are difficult to prototype and experiment with. An example of one such type of system is large-scale communication networks. Thus, they were chosen to present our ideas.

Large-scale communication networks are essentially unbounded systems, characterized by distributed administrative control without central authority. They have limited visibility beyond the boundaries of local administration, and at any individual node, have incomplete information about the network's topology and component functions. We found an emergent algorithm-based approach ideal to model such systems realistically, as they produce global system-wide properties that emerge from the collective actions of the participating nodes.

The rest of this paper is structured as follows. Section 2 describes the challenges faced in modeling and simulating unbounded systems. Section 3 describes the advantages, based on our experiences, of using EASEL (Emergent Algorithm Simulation Environment and Language), an emergent algorithm-based language, for modeling and simulating such systems. Section 4 contains a detailed description of our model of a large-scale communication network and the simulation of the distance vector routing algorithm on it. Section 5 concludes the paper, ratifying the efficacy and realism provided by the one-to-one approach to the model and simulation.

2 UNBOUNDED SYSTEMS SIMULATION: CHALLENGES AND APPROACHES

Accurate simulation of unbounded systems has been a hard to attain goal in traditional simulation. Participants in unbounded systems, whether human or computerized, have only inaccurate and imprecise information about global state. Boundaries are blurred and constantly changing in such systems between participants. Such systems have no known or effective means of central control. Systems that resemble this description are those of large-scale critical national infrastructures, the Internet and most socio-economic and biological systems (Duego, Oppacher 1993). The critical concept to understand is that unbounded systems contrast dramatically with the assumptions such as closed nature central control, which are used to model them.

An effective approach to modeling unbounded systems is emergent algorithms. An emergent algorithm is any computation that achieves formally or stochastically predicable

global effects, by communicating directly with only a bounded number of immediate neighbors and without the use of central control or global visibility. Emergent algorithms, unlike conventional algorithms, operate in the absence of complete and precise information, central control and hierarchical structure. Emergent and genetic algorithms share the characteristic of being self-stabilizing (Kutten, Patt-Shamir 1997). They also frequently share many of the environmental constraints of unbounded systems. These include direct communications only with local neighbors, absence of central control, and the inability of individual participant components to view the state of the system from a global perspective. Emergent algorithms have the potential to generate and maintain global properties in the context of unbounded systems whether or not those properties can be generated locally in individual components.

3 EMERGENT ALGORITHMS AND EASEL

Current simulation systems do not produce accurate predictions of the behavior of unbounded systems. By definition, unbounded systems are incompletely and imprecisely defined. Thus, a simulation of an unbounded system must be able to produce accurate results based only on incomplete information. However modeling in the current framework requires complete information, which introduces assumptions and inaccuracies. Equally important, all object-based models (both physical and computerized) are inherently inaccurate because they are based on complete representations as objects. This might be acceptable when dealing with small numbers of nodes or when great care is taken to differentiate between which modeling results are likely to be valid. Such remedies seldom, if ever, succeed in differentiating inaccurate results when modeling complex or large-scale systems. Furthermore, as the number of subsystems in a model increases, the inaccuracies of each subsystem pervade the whole after a few iterations and guarantee that all simulation results will be inaccurate.

This may account for the pervasive failure of large-scale simulations to produce accurate results. These problems are aggravated in unbounded systems where the numbers of components are very large and a primary purpose of simulation is to accurately predict the global effects of local activities. Because accuracy and completeness are not simultaneously achievable when describing the physical world, accurate simulation is feasible only if the simulator can guarantee accurate results from accurate but incomplete specifications.

The use of EASEL as a modeling and simulation tool for unbounded systems was an obvious choice, as it allows simulation of thousands of semi-autonomous actors cooperating in a simulated world without global visibility or central control. It was preferred to discrete event simulation languages as it supports a loosely coupled distributed network model in contrast with the latter's shared memory multiprocessor or multiprogramming models with interleaved semantics. This loosely coupled multiprocessing model with near neighbor communication with parallel semantics, as supported by EASEL (Fisher 2000), was considered more appropriate for modeling unbounded networks. The use of the Star-Logo language (Resnik 1995), which is intended for simulating emergent-like algorithms, was also considered for this purpose. However, it requires that simulations be expressed in terms of central control and global visibility, which are often in contrast with the activities they simulate.

EASEL follows the paradigm of property based types to provide accurate, but incompletely specified descriptions of objects and processes in the real world. EASEL can be used to produce accurate conclusions about the examples from the physical world in contrast to physical models or automated simulations where models are constrained to be completely specified causing results to be accurate only for the model and not its representation in the real world. While traditional modeling and simulation systems answer all questions without a mechanism for user to determine which answers are accurate, EASEL reports what additional information is needed to continue toward an accurate result.

Complete models are inaccurate. Any physical model cannot be an exact replica of the original, as in the minimal case it will differ from the original in its positional characteristics. Completeness and accuracy are contradictory goals. The latter is achieved by promoting generality at the cost of being incomplete. Computer models also are usually complete thus introducing inaccuracy. They require specification even in the absence of formal knowledge of properties. This comes out of the fact that in conventional programming languages, representation of model components is driven by the design of data structures rather than the description of the properties of the components.

EASEL is a property-based language, which requires the author to specify properties of an actor rather than their computer representations. An actor is any active entity of an EASEL program, simulation, or processor within a simulation. At the program level (but outside of simulations), actors act as multi-programmed tasks with the shared memory environment of a Macintosh application. Actors, in general, have a set of properties and behavior.

This makes the model closer to real world as describing the properties of a real world system comes more naturally to humans than their data structure representations.

EASEL, unlike other simulation languages does not reduce the problem of mapping the real world representations to data structures. This aspect is implemented by the underlying system in the most appropriate manner. The user can then concentrate on the accuracy of the description. Due to this fact, the inaccuracy in the model is proportional to the inaccuracy introduced during the descrip-

tion of properties, not the choice of data structures by the author of the simulation.

Furthermore, simulations in conventional languages like C, C++ and Java simulate the effect of an event rather than the event itself. This widens the gap between the simulation and the real world activity, thus reducing accuracy. In conventional programming languages, we would require a large number of threads to simulate the action of each independent actor. However in EASEL each actors' actions are simulated as described and the end result is that of the actual actions and not of the effects of the actions.

EASEL also separates the model from the simulation (Fisher 1999) thus increasing the reusability of models. Breaking up the problem into these two domains further simplifies the simulation, as an author of a simulation need not know the complexity of the underlying model. It also facilitates the distinction between a user (one who runs the simulation to observe trends and behavior critical to his needs) and an author (one who is a domain expert in a particular field and describes the properties and life cycle of the various components and their interactions with respect to a bounded set of neighbors).

Another distinct advantage of EASEL is that inheritance is implicit in the language as opposed to other object-oriented languages where it is explicit and has to be specified by the user. This feature further reduces the complexity for an author allowing him to concentrate on the description of the model and/or simulation rather than on the internal representations of the model components.

```
positional: type is {
    x : number := ?;
    y : number := ?;
}

move (pos : positional , delta_x : number,
delta_y : number): action is {
    pos.x = pos.x + delta_x ;
    pos.y = pos.y + delta_y ;
}

ant: actor type is {
color : …
    x : number : = …;
    y : number : = …;
}
move (ant , <value> , <value>);
```

The "ant" actor is a valid parameter to the move( ) action as it meets all the constraints required by the positional type. This is different from conventional programming languages like C and C++ where the properties of positional type should be the only ones that ant should possess in order to be acted upon by the move function. This further emphasizes the reusability of EASEL modules.

Development of models and simulations were simplified by the fact that no type conversions are needed in EASEL (Fisher, Christie 2000). This is possible because EASEL permits operations between any types. For example, an expression testing equality between a floating point number and an integer is valid without an explicit typecast. A more interesting example is as follows:

```
router : actor type is {
……
……
}

mobile_router : actor type is {
……
……
}
```

Instances of these types can be tested for equality. The equality operates on the intersection of the properties of the two actors.

The above property is especially useful in developing systems consisting of similar components with a certain degree of heterogeneity without completely redefining the components. An example of such a system is a communications network where certain routers maybe mobile and others stationary. The mobile routers can simply be defined as in the above example instead of completely redefining the properties of routers in the *mobile_router* actor type. This improves the reusability of previously defined actors.

One of the drawbacks, however, is that a statement such as **new <actor type>** triggers the EASEL translator to view the description of the actor type as specified by the author and decide a computer representation to best reflect the properties of the object. This translation results in the introduction of a certain degree of inaccuracy that is a function of model description inaccuracy.

4 NETWORK SIMULATIONS USING EMERGENT ALGORITHMS

In our model of an internetwork *routers* and *links* (between the routers) are modeled as actors. The need to model routers as actors was straightforward, as they have several inherent properties (e.g. position in the network, IP address, list of connected links, input message queue and routing table) and distinct behavioral characteristics (e.g. message sending, message receiving and message processing). Links, on the other hand, could have been modeled as properties of the router actors. However, due to their dynamic bandwidth property, they were modeled as actors. Both router and link actors act independently of each other, thus, truly simulating the real world scenario.

The routers are added incrementally to the network in a true sense; each one positioning itself in the network *view*, creating links with other existing routers as per the Waxman model (Zegura 1996) and sending control messages to their neighboring nodes advertising their presence in the network. The routers are depicted as circles centered at randomly generated x- and y-coordinates between the

origin and the maximum possible x- and y-coordinates of the view. The link creation in accordance with this model produces realistic network topologies. The routers, next, initialize their routing tables according to the Distance Vector Routing protocol (Bertsekas 1991). This task entails that each newly added router request all its neighbors' routing tables by broadcasting a *req_neighbor_dist_vector* message. The message is placed by the router on the message queue of the links, which detect its presence and forward it to the message queue of the destination router. This asynchronous message passing is accurately reflective of the real world, as it does not require an actor to wait for another. The recipient routers transmit their routing tables in the body of a *response_to_ req_neighbor_dist_vector* message to the router specified by the source address of the *req_neighbor_dist_vector* message, in a similar manner. The recipient of the response message checks its existing routing table. If for any router in the sender's routing table (i.e. in the message), there already exists an entry in the recipient's routing table, the entries are compared as per the following algorithm:

if number of hops in current entry > number of hops advertised by sender + 1 then
replace current entry for the appropriate router by new entry

However if no entry exists in the recipient's table for a particular router, for which an entry exists in the sender's table, a new entry is added in the recipient's table with the advertised number of hops incremented by 1. All the routers follow this procedure independently as and when they join the network. This is in contrast to most network simulation methodologies where a central authority controls the router creation and initialization process.

The routing of data messages (*normalmsg*) between routers was used as a means to validate the network model and Distance Vector Routing protocol simulation. The message packets were generated as per a Poisson process and destined to random nodes.

The inherent dynamism of communication networks (Paxson et al 1997) was accounted for in the simulation by adding the routers one at a time in the network. This further emphasized the lack of need for central control and global visibility in the network as each router initialized its tables and operated independent of the operation of other routers.

The simulation was carried out for a total of 100 packet transmissions for randomly selected source-destination pairs in a 50,000 node network. It was averaged over 10 runs to eliminate any discrepancies in the simulation results. Each router, on an average, experienced a load of 31 data packets and 6 control packets during the course of the simulation.

A limitation of the language worthy of mention is the finite number of routers that could be successfully supported. At the time of this simulation, EASEL could support a maximum of 75,000 actors.

However, the integral conformance of the graphical and text output of the simulation with the protocol ratified the validity and accuracy of the simulation and network model.

5 CONCLUSION

The successful modeling of an unbounded system, namely a large-scale communication network, to truly reflect the activities of the various components in the real world ratifies the feasibility of this approach. The successful simulation of the distance vector routing algorithm over this model further emphasizes the correctness of the model. The use of EASEL simplified the process of design and implementation of the model and simulation throughout the experience, as we as the authors could concentrate on the description of the properties of the various actors and their interactions rather than on the data structures to represent them in the computer. The problem was thus altered from being one of mapping the real world into computer to one of actually describing the real world, which we felt was closer to human instincts than the former.

REFERENCES

Bertsekas, G. Data Networks. Second Edition. Prentice-Hall, Dec 1991.

Duego, D. and F. Oppacher. Achieving Self-stabilization In a Distributed System using Evolutionary Strategies. *Proceedings of the InternationalConference on Artificial Neural Networks and Genetic Algorithms.* Berlin, GermanyApril 14-16, 1993.

Fisher, D. A. Design and Implementation of EASEL: A Language for Simulating Highly Distributed Systems, Dec 1999.

Fisher, D. A. EASEL Survivability Simulation System.*Impacts 2000, 15th Annual Software Engineering Symposium.* Washington DC, September 18-21, 2000.

Fisher, D. A. and A. Christie. EASEL- A new simulation language and its application to software process. *Prosim 2000.*

Kutten, S. and B. Patt-Shamir. Time adaptive Self-stabilization. *Proceedings of the 16th Annual ACM Symposium on Distributed Computing*. August 1997.

Resnik, M. New paradigms for Computing, New Paradigms for Thinking. Computers and Explanatory Learning ,A.diSessa,C.Hoyles And R. Noss,editors, Springer-Verlag (1995).

Zegura, E. W. et al. How to Model an Internetwork. INFOCOMM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies Networking the Next Generation, Proceedings IEEE, Volume 2,1996. Pages 594-602 vol 2.

Paxson, V. et al. Why we don't know how to simulate the Internet. Winter Simulation Conference 1997. Atlanta, GA.

AUTHOR BIOGRAPHIES

ROHYT BELANI is currently a graduate student in the Information Networking Institute (Electrical and Computer Engineering Department) at Carnegie Mellon University. He received his B.E. degree in Computer Science from the University of Bombay, India. His current research interests include Modeling and Simulation of Networks, Wireless Communications and Networking, and Distributed Systems. His email address is <rbelani@andrew.cmu.edu>

SAUMITRA M. DAS is currently a graduate student in the Information Networking Institute (Electrical and Computer Engineering Department) at Carnegie Mellon University. He received his B.E. degree with Honors in Electronics Engineering from the University of Bombay, India. His current research interests include Wireless Communications and Networks, Distributed Internet Services and Optical Networks. His email address is <smdas@andrew.cmu.edu>

DAVID FISHER is currently leading a research effort in new approaches for survivability and simulation in information-based infrastructures at the SEI's CERT® Coordination Center. From 1973-75, Fisher served as program manager in the Advanced Technology Program (ATP) at the National Institute of Science and Technology (NIST), where he developed and managed a major initiative in component-based software and began an initiative in learning technology. Fisher has more than 60 publications in the areas of information survivability, algorithms, component-based software, programming languages, compiler construction, and entrepreneurial development in the software industry. He earned a PhD in computer science at Carnegie Mellon University, an MSE from Moore School of Electrical Engineering at the University of Pennsylvania, and a BS in mathematics from Carnegie Institute of Technology. His email address is <dfisher@cert.org>