Proceeding of the 2004 American Control Conference
Boston, Massachusetts June 30 - July 2, 2004

An Agent-Based Simulation Laboratory for Economics and Infrastructure Interdependency

David A. Schoenwald, Dianne C. Barton, and Mark A. Ehlen

Computation, Computers, Information, and Mathematics Center
Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185-0318
daschoe@sandia.gov

Abstract

Researchers at Sandia National Laboratories have developed a next-generation agent-based economic 'laboratory' (N-ABLE) for analyzing the economic factors, feedbacks, and downstream effects of infrastructure interdependencies. N-ABLE is a simulation environment in which hundreds of thousands of individual economic actors simulate real-world manufacturing firms, government agencies, and households. N-ABLE will be demonstrated on a laptop supported by both an input and output graphical user interface that will allow users to change parameters and simulate a multitude of "shocks" to the economy including electric power outages and shipping port closures. The tool has been used by several universities and three national laboratories for studies involving the economic impact of infrastructure and supply chain disruptions.

Introduction

The most common tools currently used for economic modeling are either macroeconomic or computable general equilibrium (CGE) tools. Both rely on regression analysis of aggregate data to develop parameters for use in economic forecasting. These macromodels can provide accurate forecasts, but problems arise when totally new economic policies are introduced with no past relevant data that can be used to develop modeling parameters. Also, modeling of shocks or discontinuities imposes certain limitations on the analysis. Although microsimulation could offer more modeling flexibility, it had been hampered by both a lack of essential microdata and the extensive calculation time required to micromodel something as large and complex as the US economy. Advances in massively parallel techniques for solving complex modeling problems, coupled with new sources of microlevel data, have now made microsimulation a viable economic analysis tool.

N-ABLE microsimulates the economy using an agent-based discrete-event model. This modeling approach is well suited for investigating the behavior of complex, nonlinear stochastic systems like the economy. Agents start each time increment making decisions much like their real-life counterparts. Decisions on actions to take are based either on probabilities computed from actual microeconomic data or on results of learning models including genetic algorithms. These decisions include purchasing products, hiring workers, selling bonds, collecting welfare payments, conducting open market operations, and others. Macroeconomic variables, such as gross domestic product, inflation (CPI), and the unemployment rate are computed as aggregate results of innumerable decisions by the individual economic agents.

NABLE can specifically address "what-if" questions such as:
1. What economic sectors are most vulnerable to infrastructure disruptions and interdependencies?
2. Which sectors have different usages of energy, transportation, financial, communication sectors?
3. What short-run economic changes affect infrastructure performance?
4. What firms are most affected? Who does well, poorly?
5. How do firms, individuals, and other economic components respond, over time and over regions?
6. What economic mechanism do national, state, and local governments have to assist firms and other economic sectors in their regions?

This analysis capability provides a rich education environment for students and researchers alike to study the impact of various policy decisions as well as learn more about the dynamics of a complex system such as an economy and infrastructure networks.

Individually and collectively, we depend on critical infrastructures in the United States to provide the essential services that support (among other things) our economic prosperity, quality of life, and national security [1]. These infrastructures have recently been categorized into the following sectors: agriculture, food, water, public health, emergency services, government, defense industrial base, information and telecommunications, energy, transportation, banking and finance, chemical industry and hazardous materials, and postal and shipping [2]. While historically these infrastructures have been vulnerable to malevolent acts and to natural disasters, today, these systems are more threatened by the increasing complexity inherent in their growing interconnectedness and interdependence [3].

Increasing interconnectivity is being driven by modern business trends that rely heavily upon telecommunications for management and operation. On balance, interconnectivity will improve our nation's economic efficiency; however, tight coupling between infrastructures also results in situations where a disturbance in a formerly isolated infrastructure unexpectedly cascades across diverse and seemingly unrelated infrastructures. In simulations using Sandia National Laboratories' Aspen Electricity Enhancement model, i.e. Aspen-EE, this effect is observed when policy decisions lead to power disruptions that cause second-order impacts on pricing trends [4]. The interconnection of these infrastructures creates tremendous interdependency and vulnerability.

Because interdependent infrastructure networks are complex systems, they do not readily submit to traditional reductionist analysis where an individual infrastructure would be examined in isolation from other infrastructure elements. The reductionist approach ignores that linkages between infrastructure elements do exist and that such linkages cannot be deduced from isolated analysis. Complex systems exhibit a rich variety of behaviors, including many that are counterintuitive. To capture these effects, we must view complex systems from a holistic rather than a reductionist point of view.

Agent-Based Approaches for Complex Systems

To analyze interdependent infrastructure systems in a more holistic way, Sandia and other research institutions have developed models of critical infrastructure systems using agent-based approaches. Sandia's first agent-based model of the U.S. economy, developed in the mid-1990s, is called Aspen [5]. This model is a Monte Carlo simulation that uses agents to represent various decision-making segments in the economy, such as banks, households, industries, and the Federal Reserve. An agent is a computational entity that receives information and act on its environment in an autonomous way; that is, an agent's behavior depends at least partially on its own experience. Through the use of evolutionary learning techniques, Aspen allows one to examine the interactive behavior of these agents as they make real-life decisions in an environment where agents communicate with each other and adapt their behaviors to changing economic conditions, all the while learning from their past experience. In 2000, Sandia developed a new model of infrastructure interdependency called Aspen-EE. This model extended the capabilities of Aspen to include the impact of market structures and power outages in the electric power system, a critical infrastructure, on other infrastructures in the economy [4].

One of the limitations of recent agent-based models is that communication is treated simply as message passing between agents. Effectively, the telecommunications infrastructure is not specifically represented. None of the models simulates the differences in communication over telephone, computer, wireless, or other networks and therefore cannot model the impact of specific communication failures on the whole system. Nor can current models simulate the impact of other infrastructure failures on telecommunications.

To address the communications deficiencies described above, Sandia revised and restructured the Aspen-EE model to include a more realistic representation of the telecommunications infrastructure. This new model of infrastructure interdependency is called NABLE. In NABLE, communication is treated as an integrated agent system capable of creating, transforming, sending, receiving, and storing information and messages over time and across distance. With NABLE, we can model communication networks or medium-specific vulnerabilities to failures and their dependence on supporting infrastructures like power.

The Model

In building NABLE, we used the agent-based model Aspen-EE as the foundation and completely revamped its architecture. NABLE was written in C++ to run on a single-processor-machine. A version of NABLE for parallel-processor machines is underway.

Agent-based models assume that complex behavior emerges from many individual, relatively simple interactions rather than from the complexity inherent in any particular agent. Agents have simple rules of behavior and react to their environment (i.e., the other agents and any static features) without reference to any global goals—in other words, the agents are undertaking purely local transactions. The net results of these local interactions and decisions are phenomena that emerge on a global level. When unexpected results emerge from the simulation, it is important to be confident that we understand the fundamental processes built into the model. The complexity of agent-based modeling should be in the results of the model and not in its assumptions.

Representing Infrastructures

There are several ways that we can implement the notion of infrastructures in NABLE. One method of representing certain types of infrastructures in NABLE is through the use of *spigots* and *sinks*. Such infrastructures are for commodities that run continuously, like water from a municipality and electricity from a local utility. A sink is where a producer puts product into an infrastructure. For example, a power company may have a natural gas-fired electric generating plant producing power. It would put power on the transmission lines by passing the power into the associated sink. A spigot is where a consumer gets the product, such as turning on the lights in a residence or getting water from a faucet.

The actual creation of sinks and spigots is hidden from the user. An agent that buys power, for example, from a power market, automatically gets a spigot from which it can draw the power. An agent that buys water from a bottled water company, however, gets shipments of bottled water, not a spigot.

Sinks and spigots serve to communicate demand and availability between a producer and a consumer. Relative to electric power, for example, the consumer agent is restricted by a power outage since they will not be able to make purchases electronically during an outage. Similarly, it is important for the producer to know that the cumulative pull on the power system at a given moment is greater than its ability to produce, so that the NABLE program can know when to "start" the outage.

Another method for representing infrastructures is through the construction of a network; such as we have done with the communications network in NABLE. If communications systems are overwhelmed with the sheer volume of communication, they could ripple through the system and, in effect, cause an outage.

The Mechanics of NABLE

Agents in NABLE, like in previous Aspen models, are decision makers. Each agent behaves the way its counterpart in the real world would behave, as the simulation traces the agent's daily actions, e.g., buying commodities, selling commodities, paying for commodities by check, credit card, or other electronic payment means, etc. Agents of the same type draw from the same decision rules. For example, all firm agents of the same type, e.g., Firm_B, use the same rule to decide from which agent that they will purchase commodities. However, the decision from which agent to actually purchase a commodity may vary from agent to agent because of its own constraints, such as insufficient income to purchase the commodity desired at a particular point in the simulation or because an agent is more concerned about getting the lowest price and spends more time shopping.

As explained previously, the user builds agent types with a set of building blocks, or templates, each with its own name. The number of individual agents of each type created during a simulation is also specified by the user. For example, a simulation in which the user creates two types of firms, Firm_A and Firm_B, might contain 10 individual Firm_A agents and 20 individual Firm_B agents.

A simulation in NABLE is a sequence of *events*. Fundamentally, an event is just a point in the sequence where something "interesting" happens. Therefore, an event is not an *action*—an event is more like a piece of paper from a "take-a-number" paper dispenser that one might find at the Motor Vehicle Division (MVD). That is, an event gives a priority to an abstract concept so that it can be sorted and scheduled. To use the MVD example, your business (which is perhaps a driver's license renewal) is an abstract concept that can be sorted and ordered simply by giving you a piece of paper with a number on it. The order in which MVD customers are processed is then given by an ascending sequence (1, 2, 3, …). That is to say, as a customer, you have been given a numerical priority that is independent of your business and one that roughly corresponds to the order in which you arrived at the MVD office (that is, it's a "fair" priority for the particular situation). In NABLE the prioritization scheme is a bit more complicated (i.e., we also prioritize by your "business"), but the concept is the same. The simulation is accomplished by processing the events in a correct sequence.

There are several types of events; all have important roles in producing the final output of the model. Two kinds of events, however, do most of the work: task events and message events. Both event types involve model computations. A task event is one in which an agent independently begins a new sequence of actions. A message event is one in which an agent communicates with another agent, possibly to complete a sequence of actions. Task events have higher priority than message events.

A task event usually starts off a chain of message events. For example, an agent can decide to pay all its bills on the first of the month. So every month, it schedules a task, "Pay All Bills" to remind itself to pay its bills the next month. Then, when the time comes the next month to "Pay All Bills," the agent sends off a series of messages along the lines of "I am paying my bill; here is a check for $50." These messages are scheduled on the calendar (see below) for the moment when they should arrive, say, four days in the future for a check sent through the postal system—and the moment they arrive is triggered by a message [delivery] event scheduled specially for that purpose.

NABLE uses the concept of a calendar to sequence events during a simulation. Events are scheduled on the calendar (a priority queue), with different priorities assigned by NABLE programmers to different events. Priority is determined by the event's time, its type, and any applicable secondary priority. At the top of the calendar, which changes dynamically, is either the highest priority event, or one of several events that have equivalent priority, where that priority is the highest priority relative to all other priorities of events on the calendar.

Time in Aspen-EE is divided into a minimum time step that is determined by NABLE users. Thus, one time step can be set to be equivalent to one minute, and any other unit of time can be derived from that. For example, if the minimum time step was one minute, as it is in the current version of NABLE, an hour would take 60 time steps, a day would take 1,440 time steps, and a week would take 10,080 time steps. Each time step can have zero or more events. Since events are prioritized by time, type, and secondary priority, one can interpret the priority scheme as "dividing" the time step into stages. For example, all task events have higher priority than any message event; hence, all task events will be processed before the first message event.

Commodity Sales and Prices. Agents will sell a certain amount of products per day. To simulate how the agents, like firms, set prices for their commodities, NABLE uses a genetic algorithm learning classifier system (GALCS) in which the agents determine four trends daily: (a) whether the commodity price has been recently increasing or decreasing, (b) whether sales have been recently increasing or decreasing, (c) whether profits have been recently increasing or decreasing, and (d) whether prices are higher or lower than the industry average. Based on answers to (a) through (d), the agent finds itself in one of 16 states.

The GALCS assigns a probability vector (p^{D}, p^{I}, p^{C}) to each state,

where p^{D} = probability that the agent will decrease a given price (by a certain exogenously specified amount) the next time the agent enters the same state,

p^{I} = probability that the agent will increase the price, and

p^{C} = probability that the agent will keep the price constant.

Upon entering a certain state, the agent decides how to change a given price by using the corresponding probability vector and choosing a random number. The agent then adjusts the vector according to how the price change affects profits. The example below can help to explain this process.

Suppose that at a particular time for state 2, the following condition exists:
$(p^D, p^I, p^C) = (0.1, 0.6, 0.3)$. Assume that an agent then enters this state and draws a random number indicating the need for a price increase. Suppose further that as a result of increasing the price, profits drop. To reflect this drop, the vector is then adjusted to (0.15, 0.5, 0.35). Thus, NABLE simulates the agent's learning process. The agent learns that raising prices in state 2 was detrimental. As a result of an incorrect decision, the vector is adjusted to reflect a decreased probability of a price increase. The changed probability vector reflects the unlikelihood that the agent will increase prices upon re-entry into state 2.For more details on GALCSs and GALCS results from other Sandia runs, see *Aspen: A Microsimulation Model of the Economy* [5].

Buyers

Any CommAgent with a Buyers block can purchase commodities from other agents who sell those commodities. A Buyers block tells the agent how to go about obtaining what it needs. Essentially, then, buyers are a way to obtain raw materials for the production.

Structure of the Communications Network for Financial Transactions

A new communications network has been developed for NABLE. No previous Aspen model contained this type of structure. The purpose of the communications network is to represent more realistically the process by which agents communicate with each other across electronic media. The traditional message-delivery system, used in NABLE and in previous Aspen models, features instantaneous receipt of a message once that message is sent. With the new network, delays are built into the process. Importantly, the new communications network does not supplant the traditional message-delivery system. Instead, it is used in tandem with that system to currently handle only financial transactions that are used for the sample telecommunications and banking problem.

Composition of the Network

The central hub of the communications network is a global router, a special agent in NABLE. The router is connected to CommTerminals (communications terminals) that belong to the individual CommAgents in the simulation, forming what is known as a star topology (see inset to right).

Agents use their CommTerminals to send messages to other agents. In the current version of NABLE, each message is sent in a single packet. A packet sent from one agent to another does not go directly to the other agent. Instead, the packet is transmitted through the global router. The router accepts as many packets as it has room for in its buffer and drops the rest. There is a prioritization scheme; basically, higher priority messages get "first dibs" on an empty buffer slot, but do not bump a lower-priority message from a filled slot. The packets sit in the router's buffer for a small randomized amount of time and are ultimately forwarded to their final destination.

The only characteristic in this entire communications system that the user can currently affect through the input file is the size of the buffer in the global router. Other characteristics can be changed to simulate outages, but this is currently done through the Router code by NABLE programmers. This capability will be added to the input file variables in future versions of the model. The buffer-size input parameter is called BankRouterBUFSIZE, which is located in the Run Data portion of the input file. With one router forwarding packets and the user having some measure of control over the number of packets the router can handle at one time, we can analyze the router's behavior in detail to ensure it is behaving correctly.

Example

To test the new communications network in NABLE, we have chosen to model interactivity and interdependency between the telecommunications and banking infrastructures. The sequence below provides further details for the various steps in the process. The term *router* in these steps refers to the global router.

1. One agent (in this case, Consumer) sends a check to another agent (here, it is Firm_B). The check writer could have an account in the same bank as the agent to whom the check is written or an account in a different bank; hence, the presence of Bank_A, which in the example could belong to the consumer. The check must go through the router.
2. After handling the check according to its message-processing rules, the router sends the check to Firm_B.
3. Firm B sends the check to Bank_B, its own bank. The check must go through the router.
4. After handling the check according to its message-processing rules, the router sends the check to Bank_B.
5. Bank_B credits Firm_B's account. If the consumer has an account at Bank_B, Bank_B will debit the consumer's account.
6. Bank_B sends a clear-check message to itself if the consumer's account is at Bank_B. If the consumer's account is at Bank_A, Bank_B sends the clear-check message to Bank_A. The clear-check message must go through the router.
7. After handling the clear-check message according to its message-processing rules, the router sends the message either to Bank_B or Bank_A.
8. If the clear-check message is received by Bank_A, that bank debits the account of the consumer.
9. At some later point, Bank_B reconciles the account with Firm_B and with the consumer if it also has an account with Bank_B. Otherwise, Bank_A will reconcile the consumer's account. The reconciliation activities are also done through messaging, but the messages are not passed through the router.

The delays introduced by the router after steps 1 and 3 prevent Firm B from being able to spend the funds. The delay in the router as a result of step 6 has little effect, but it adds traffic to the router. By introducing a delay for step 9, the process prevents both the consumer and Firm_B from knowing whether any deposited checks (from step 5) have arrived, i.e., effectively cleared, and that the associated funds are ready to be spent.

The approach taken in the check-clearing process enables banks to make the funds from all deposited checks (even for very large amounts) available immediately. This policy decision can be changed, but it adds more steps to the whole transaction. The funds-availability policy is really independent of the underlying network structure.

Results and Analysis

One basic test problem was used in all of the analysis runs. The problem is a supply chain with Consumer agents driving demand for goods at the end of the chain. For example, type F firms require two units of A and two units of C to make 1 unit of F. There was also one Bank agent in the problem. Various individual parameters such as production level and router buffer size were altered for each test run. Simulations using the model were run with the following individual agents:100 Consumer agents, 10 Firm_A agents, 10 Firm_B agents, 7 Firm_C agents, 4 Firm_D agents, 3 Firm_E agents, 3 Firm_F agents, and 1 Bank agent.

Analysis of the Router Buffer Capacity

As a first analysis, we perturbed the size of the finite communications buffer and evaluated the impact on the percentage of dropped packets. An analysis was done for a buffer size of 7, 9, 10 and 12. At a buffer size of 12, there were no packets dropped during the simulation.

We would expect the number of dropped packets to increase supralinearly as the buffer size is reduced, and that is observed in the test-case simulation. We define here the concept of severely errored time steps (SETs) by analogy to the term *severely errored seconds* used by the telecommunications industry. Severely errored seconds is a measure of the degradation of transmission lines [6]. Here, a SET is a measure of the degradation of a packet connection between agents, and is defined as the percentage of dropped packets in a time step that exceeds a particular threshold. Figure 1 presents a histogram of SETs for each of the three buffer sizes, as the threshold in the definition of SETs is varied. The increase in dropped packets when the buffer size is reduced from 9 to 7 is significantly greater than twice the increase in dropped packets when the buffer size is reduced from 10 to 9.

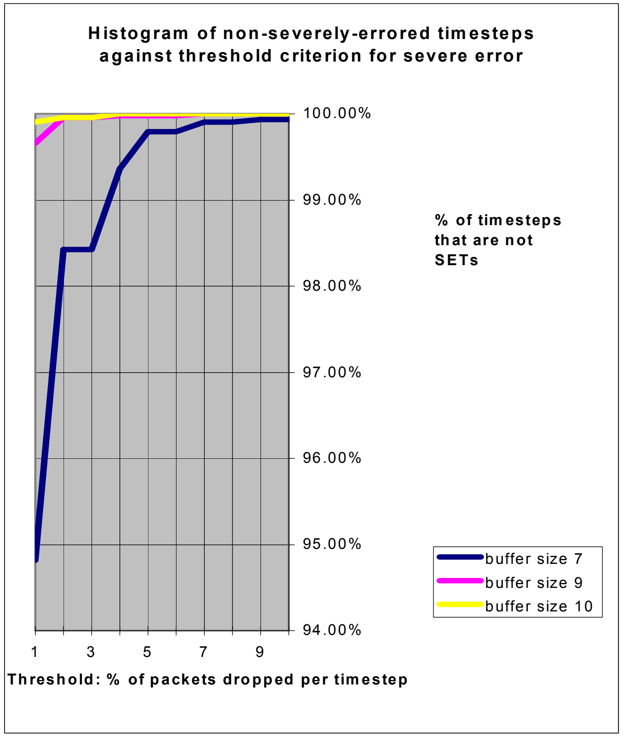

Figure 1. Histogram of SETs for the three buffer sizes.

For example, for a SET threshold of 10% packet drop, 99.93% of the time steps are not SETs for a buffer size of 7; whereas, no time steps are SETs for buffer sizes of 9 and 10. For a SET threshold of 1% packet drop, 5% of the time steps are SETs under a buffer size of 7, 0.3% of the time steps are SETs under a buffer size of 9, and 0.1% of the time steps are SETs under a buffer size of 10. These results suggest that there are decreasing marginal benefits for increasing the size of the communications buffer, as would be expected.

Analysis of Router Outage

The router in this scenario is the communications router for the Bank agent. The router has a variable buffer size that was varied in a range of 7 to 14 packets. Once the buffer is full, the router no longer accepts new packets. These packets are dropped by the router. The agents that sent these packets (e.g., households, firms, etc.) may resend at another time or abandon their transmission.

The purpose of this analysis was to study the impact of an outage on the percentage of packets dropped by the router following resumption of communication services. The outage was tested as both one week and two weeks in duration. Aside from the length of the outage itself, there was not a significant difference in router performance following the outage as a function of the one- or two-week outage. There was more variation in performance as a function of buffer size. For buffer sizes greater than or equal to 10, there were no dropped packets prior to the outage. This was expected since the larger buffer size is sufficient to accommodate the communications traffic generated by all the agents used in the NABLE test case. For buffer sizes of 9 or smaller, there are occasional dropped packets that increase as the buffer size is lowered to 7. For buffer sizes of less than 7, there are significant packet losses due to the small buffer size, which is not sufficient to accommodate the communications traffic in the test case.

It is postulated here that the communications terminals of the NABLE agents have stored up messages they wish to send and have held them in reserve waiting for the router buffer to clear. Thus even after the router buffer clears, the agents will continue to send messages that had been dropped during the outage or post-outage transition.

This behavior of resending messages implies a memory effect among the agents that exceeds the memory effect of the router itself. Thus, the continued nonzero packet losses following the clearing of the router buffer after an outage is explained by this postulate. There may be means to handle this situation differently in the future. For instance, the agents could be put on an organized schedule following an outage. This schedule would dictate when agents can resubmit their dropped messages as well as their new messages for some period of time following an outage. This schedule could be determined by prioritizing the agents (perhaps certain businesses would have a higher priority than households, etc.) and also by the size of the router buffer. As the router buffer size is increased, this post-outage behavior is improved, but one could argue that traffic will always exceed any buffer's ability to handle it following a long enough outage. Still, one lesson of this analysis is to use as large a buffer size on the router(s) as is economically feasible. A second lesson is to employ a more organized post-outage message-resending behavior perhaps based on a schedule or some optimization criterion. A third lesson is some kind of fail-safe communications between the router and the communications terminals, i.e., CommTerminals, that make it clear that there is an outage that is not immediately repairable. This may then trigger the scheduling protocol proposed above. Clearly, the more sophisticated schemes the router and communications terminals can employ when dealing with an outage, the more quickly and smoothly the network will recover from an extensive outage. Work has been done to mathematically model these finite buffer queues such as the one simulated in NABLE. Reference 7 describes a queuing network model for finite capacity queues such as is realistic in electronic finance networks.

Acknowledgement

This work was supported in part by the Laboratory Directed Research and Development (LDRD) program at Sandia National Laboratories and in part by the National Infrastructure Simulation and Analysis Center (NISAC), which is funded by the U.S. Department of Homeland Security. Sandia National Laboratories is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

References

1. Sandia National Laboratories. *U.S. Infrastructure Assurance Strategic Roadmaps: Strategies for Preserving Our National Security*, SAND98-1496. Albuquerque, NM: Sandia National Laboratories, 1998.

2. U.S. Department of Homeland Security. *The National Strategy for the Physical Protection of Critical Infrastructures and Key Assets*. February 2003. Available at http://www.dhs.gov/interweb/assetlibrary/Physical_Strategy.pdf (accessed 13 September 2003).

3. Barton, D. C., and K. L. Stamber. "An Agent-Based Microsimulation of Critical Infrastructure Systems." In *Global Energy Exposition 2000 Proceedings*, held in Las Vegas, NV, July 2000. Lancaster, PA: Technomic Publishing Company, 2000.

4. Barton D. C., E. D. Eidson, D. A. Schoenwald, K. L. Stamber, and R. K. Reinert. *Aspen-EE: An Agent-Based Model of Infrastructure Interdependency*, SAND2000-2925. Albuquerque, NM: Sandia National Laboratories, 1998.

5. Basu, N., R. J. Pryor, T. Quint, and T. Arnold. *Aspen: A Microsimulation Model of the Economy*, SAND96-2459. Albuquerque, NM: Sandia National Laboratories, October 1996.

6. "RFC INDEX." Available at http://rfc-1232.rfcindex.com/rfc-1232-5.htm (accessed 3 October 2003).

7. Balsamo, S., V. De Nitto Personè, and P. Inverardi. "A Review on Queueing Network Models with Finite Capacity Queues for Software Architectures Performance Prediction." *Performance Evaluation* 974 (2002): 1–20.