Network Survivability Analysis Using Easel

Alan M. Christie

December 2002

TECHNICAL REPORT
CMU/SEI-2002-TR-039
ESC-TR-2002-039

Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

Network Survivability Analysis Using Easel

CMU/SEI-2002-TR-039
ESC-TR-2002-039

Alan M. Christie

December 2002

Networked Systems Survivability Program

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

(signature on file)

Christos Scondras
Chief of Programs, XPK

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Table of Contents

List of Figures

List of Tables

Executive Summary

Given today's concern about the survivability of large complex networks such as the Internet, there is a need to understand how these systems respond to planned attacks or random accidents. This understanding requires insights into how information or artifacts flow through networks and how networks respond to major disruptions. This in turn requires knowledge of how nodes in the network are linked together, how they communicate, and how the failure of selected nodes affects the performance of the network as a whole. Significant work in this area is ongoing (see review in Appendix A) but a full understanding has yet to be achieved. Because the networks of interest can be topologically complex, are highly non-linear in their responses, and are inherently unbounded (i.e., no node in the network can have global knowledge), they are difficult to analyze. These issues are addressed in this report, which provides practical techniques for modeling networked systems and illustrates the use of these techniques with examples.

The basis for these explorations is the Easel modeling and simulation language. This language is a general-purpose programming language that has enhancements for performing simulation of networked systems. Because Easel is not yet widely known, we first review some of the issues that motivated its development (e.g., emergence in unbounded systems) and describe some of the key features of the language.

Since network topology is a central theme to the report, we investigate how large networks can be synthesized. In this regard GENeSIS, a program written in Easel, is described and used to illustrate the construction of network topologies. Networks with different topological properties can be built to examine, for example, their relative survivability. The GENeSIS program provides support to output topologies that can be used in survivability and other network applications.

As an example application of synthetically generated networks, we investigate the propagation of computer viruses. Two virus propagation models are defined and use networks produced by GENeSIS as input. The models are simple but the exercise results in complexity because of the non-deterministic manner in which the virus propagates through the network. A variety of parametric variations are examined. For example, we look at the sensitivity of the number of compromised nodes relative to the length of time before a virus patch is released.

Abstract

The survivability of large, complex networks such as the Internet is an increasing concern, but such networks are difficult to analyze because they are topologically complex, highly non-linear in their responses, and inherently unbounded (i.e., no node in the network can have global knowledge). To support survivability research, this report will describe how to develop statistically valid networks for analysis and, as an example of their use, applying them to the simulation of virus propagation. It will illustrate the construction of network topologies with GENeSIS, a program written in the programming language Easel. The report will also summarize ongoing significant work in this area of research and give readers insight into how information or artifacts flow through networks and how networks respond to major disruptions.

1 Introduction

This report describes the results of explorations into the use of simulation in examining Internet survivability. As originally conceived during the Cold War [Baran 01], the ARPANET was driven by survivability requirements, but these early notions were not sustained. In particular, when the Internet was formally defined in 1995,[1] issues of survivability became a distant memory. However, given the Internet's exponential growth, its importance to commerce and the military, and the increased concern about terrorism after September 11, 2001, Internet survivability is now a critical issue. Survivability is a central theme of the analyses reported here.

Networks such as the Internet have global reach with no central control. This lack of central control may make life harder for security personnel (or a paranoid government), but it does help survivability. In particular, the Internet has no single point of vulnerability that, if attacked, can bring the system down. However, the Internet is still vulnerable to attacks due to weaknesses in its topological structure [Albert 02b], its response to malicious code [Staniford 02], and its lack of robust implementation (distributed denial of service or "DDoS" attacks, buffer overflow, etc.).

The Internet's spatial and temporal characteristics can best be described stochastically, as no "grand design" is imposed. These stochastic properties arise out of the opportunistic growth of subnets connecting to the Internet and the myriad local actions that take place every second (e-mails, Web downloads, etc.). From this state of dynamic flux, stable stochastic properties arise in terms of both network linking patterns and message transmission patterns. These properties are called emergent, as they are not designed into the system but result from aggregation of local interactions, and often have a strong bearing on the network's robustness in the face of attacks and accidents. Network linking patterns are the focus of the first topic of this report, while virus propagation is the focus of the second.

The ability of the Internet to survive attack is strongly dependent on the manner in which its nodes (routers and other hosts) are connected together [Albert 02b]. As will be discussed later, it has been empirically demonstrated [Faloutsos 99] that the number of links associated with the nodes follows a power law. This law states that the probability of a node having a specified number of links (called link degree) is proportional to that number of links raised to a constant α (α~2.1). Because this distribution has no characteristic peak in link degree, it is sometimes

[1] See http://www.isoc.org/internet/history/brief.shtml.

called “scale-free.” The distribution is also sometimes called “heavy-tailed” because a small percentage of nodes have a disproportionately large number of links.

While the Internet topology is relatively robust against random failures (as evidenced by experience), this is not true for attacks, since routers with high link degree make prime targets. Taking down these routers would result in severe bottlenecking or even worse, isolation of parts of the Internet. This raises a question: Can we replace highly-linked routers with a number of less-linked routers, while providing the same quality of service and incurring an acceptable economic penalty? In addition to being an important topic in its own right, the ability to generate synthetic Internet topologies is an important foundation for many other investigations into Internet survivability. In particular, an illustrative application discussed in his report (virus propagation) depends on accurate modeling of the Internet’s topological properties.

In virus propagation, emergent properties again arise. When a virus is released, the manner in which it propagates over the Internet is difficult to predict. It may appear to be dormant for some period of time only to flourish later, or it may make a vigorous entry and then die down quickly. Theoretical analysis of viral propagation can be very insightful [Frood 02, Pastor-Satorras 02b], but simplifying assumptions often mean that real-world complexities have to be ignored. Thus simulation is a key contributor in helping understand or predict the Internet’s response to virus attack.

2 The Easel Modeling and Simulation System[2]

Easel is a conceptually new language and is central to the simulations that are to be described. This section provides a brief overview of the characteristics of the language. We will review some of the ideas that motivated the development of Easel; for example, its ability to analyze and understand complex systems that are inherently survivable. To this end, we will look at the notions of emergent algorithms and unboundedness and how they have influenced Easel's design. Further details on Easel's features and implementation can be found at the Easel Web site.[3] However, a detailed knowledge of Easel is not required in order to understand subsequent sections of this report.

2.1 Unbounded Systems and Emergent Algorithms

Easel is a language designed to model systems where unboundedness and emergence are central themes. An unbounded system is any system in which the participants (human or computerized) have only incomplete or imprecise information about the system as a whole. They include human participants as well as automated components. Their boundaries are not precisely known. Interconnections among participants in unbounded systems change constantly. Furthermore, the trustworthiness and often the identity of participants is unknown. Centralized administrative control cannot be fully effective in such systems. These are the characteristics of critical national infrastructures, the Internet, and electronic commerce. They characterize most social, economic, and biological systems, and most activities one participates in every day. Such systems contrast dramatically with the assumptions of closed, centrally-controlled computer systems and with the assumptions underlying many modern computer security technologies.

Emergent algorithms differ from conventional hierarchical and distributed algorithms: they operate in the absence of complete and precise information; do not have central control, hierarchical structure, or other single-point vulnerabilities; and achieve cooperation without coordination. Mission requirements are satisfied in the form of global system properties that emerge from the combined actions and interactions of all system components. For reasons of mission survivability, our research considers only emergent algorithms that do not have single (nor any fixed number of) points of failure. For reasons of practicality and affordability, we

[2] The author wrote this chapter with David Fisher and David Mundie.
[3] See http://www.cert.org/easel/.

consider only those emergent algorithms in which the cost of each node (whether measured in dollars, CPU cycles, storage requirements, or communications bandwidth), is less than proportional to the number of nodes in the system. The effectiveness of this approach can be further enhanced by dynamic trust validation among the participants.

2.2 The Need for Emergent Simulation

Although the benefits of ad hoc development of emergent algorithms has been demonstrated, a rigorous process for deriving emergent algorithms from mission requirements is a prerequisite to their widespread use in automated systems. Intuitions about the global effects of local actions and interactions among large numbers of nodes are seldom correct. The problem of designing emergent algorithms is especially difficult, because it begins with the desired global properties and attempts to determine which simple combinations of local actions and interactions would produce those effects over time in a large-scale network. An effective design methodology will depend on greater understanding of the influences of local action and interaction on emergent global properties and on the sensitivities of emergent properties to local variations.

The obvious and probably only means to answer these questions is by simulation of emergent algorithms and the unbounded systems in which they operate. This recognition has opened a new area of research for simulation of unbounded systems. 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. Current models, however, require complete information and thus are always built with assumptions or inaccurate information. The ability to operate on abstract specifications and simulate at various levels of abstraction is a long-standing need of many applications, but is not provided as a feature of existing simulation systems. 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 from 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. Other difficulties in simulating unbounded systems include the following:

- the need for thousands to tens of thousands of nodes per simulation
- the lack of linguistic mechanisms in programming languages for making incomplete and imprecise specification
- the inability of object-oriented computations to describe and abstractly reason about the real world
- the need to combine information about a system from multiple knowledge domains
- management of multiple simultaneous beliefs of the various stakeholders in an infrastructure
- integration among separately developed simulations
- quadratic increases in computational cost that accompany linear increments in the granularity or number of nodes in a simulation (the so-called n-squared problem)

2.3 The Easel Solution

These considerations have led to a new approach to simulation: an emergent algorithm simulation environment and language called “Easel.” Easel employs a paradigm of property-based types (i.e., abstract classes of examples described by their shared properties) which have the ultimate goal of addressing all of the above simulation problems. Because Easel is property-based it can be used to give accurate, though incomplete, descriptions of anything in the physical world. In combination with an automated logic system that has yet to be developed, it will be used to produce accurate conclusions about examples from the physical world. This contrasts with physical models and automated simulations that depend on representation of objects, where descriptions must be complete (and thus inaccurate) and in which conclusions are accurate only for the model but never for their extensional interpretation in the real world.

Easel is currently a discrete event simulation language with limited support for continuous variables. It supports multiple levels of abstraction, multiple simultaneous belief systems, distributed specification, and dynamic graphic depictions. By using quantifiers, adjectives, improper nouns, pronouns, and other forms of anonymous reference, Easel overcomes the linguistic limitations that impair traditional programming systems’ ability to handle incomplete and imprecise descriptions. In combination with property-based types, these mechanisms provide a semantic framework of examples of any type, whether real or imagined, and whether from the computational, mathematical, or physical worlds.

Thus we believe that concepts of local action and visibility, unboundedness, and emergent properties are important factors in simulating systems where large numbers of loosely coupled actors are involved; to this end, we have developed a new general-purpose simulation language called Easel, which is currently being implemented. The rest of this section summarizes the concepts behind Easel, describes some of its unique properties, and provides a simple example of Easel’s use.

2.4 A Type Is a Type Is a Type…

Being a property-based type (PBT) language, Easel considers all entities to be types. A type is a description of some class of objects, while a description is a set of properties. An example of a type is any object that satisfies the type's properties. Easel has a built-in family of types that can be extended to specify user-defined types.

Types are built up by inheriting properties of parent types. The following are some characteristics of Easel types:

- All types inherit from the root type "all." This includes all consistent (true) types and inconsistent (false) types.
- The type that contains all types is the "false" type. It is so named since types with inconsistent properties (e.g., 3 > 5) are also types. However, the false type has no examples.
- As one ascends the type hierarchy, types accumulate more and more examples from their children.
- As one descends the type hierarchy, types accumulate more and more properties inherited from their parents.

Here are some representative Easel types:

- Mutable – a type whose properties are changeable
- Immutable – a type whose properties are unchangeable
- Actor – a physical "thing" that has behavior and is threaded
- Abstract – a type that can be described completely within computer memory
- Type – the set of named types is also a type (the type type)

Even the number 5 is a type, albeit a degenerate, singleton type; it inherits properties from the positive integer type and odd integer type, among others.

2.5 Type Manipulation Within Easel

The Easel type hierarchy allows you to build up and manipulate types. This is unlike other conventional programming (or simulation) languages where you can only operate on examples of the type (e.g., through iteration). In the object-oriented paradigm, classes are equivalent to types. However, Easel's types can be manipulated in powerful symbolic ways that are not possible with classes. The program shown in the example below illustrates some of the ways that types can be manipulated. (Note that this is a program, not a simulation.)

```
dog_gonned( ): action is
   food: type is enum(meat, vegetable, anything);
   dog: type;
   omnivorous: type is
      diet :: food := anything;
   Corgi: type is
      property dog & omnivorous;

   Jenkins: Corgi;
   confirm Jenkins isa omnivorous dog;

dog_gonned();
```

As is usual in Easel, we first define some simple types, and then build up more complex types that inherit the properties of these simple types. Thus we begin by defining the type food, which has vegetable, meat, and anything as values, and the type dog, which has no specified properties. Then we build up the adjectival type omnivorous, which is the type of all creatures with a diet of anything, and the type Corgi, which is an omnivorous dog. Finally we declare Jenkins to be an attribute whose type is Corgi, and confirm that he is an omnivorous dog. Easel uses **:** to define constants and **::** to define attributes whose value may vary over time.

Parameterized types allow subtypes to be defined in the same way that adjectives qualify nouns in English.

```
example( ): action is
   flower: type;
   red(any): type;
   large(any): type;
   rose: type is large red flower;
   American_Beauty: rose;
   confirm American_Beauty isa red flower;

example();
```

Here “large red flower” is an adjectival phrase that returns the subtype of flowers that are large and red. This use of adjectives provides Easel with a powerful mechanism for defining subtypes.

In a similar way, the use of quantifiers such as any, all, some, and numeric quantifiers provides Easel with a means of selecting subtypes that have to be manipulated or tested.

```
example( ): action is
   flower: type;
   red(any): type;
   large(any): type;
   bouquet: list flower := 3 new large flower;
   biggy: type is all large flower;

example();
```

Here “3 new large flower” is a quantified expression that produces a list of three large flowers.

2.6 Actors and Neighbors

Easel’s architecture is designed so that it can simulate very large numbers of independent actors. Actors are simulated entities of the physical world (e.g., a system administrator, user, intruder, automobile, bird, or the moon), the electronic world (e.g., a computer, router, or peripheral device), or the software world (e.g., a software agent or task). Giving each actor its own thread of control allows for a high degree of parallelism in Easel’s execution. Actors can interact directly only with their near neighbors, and only in ways prescribed by their neighbor relationships. Neighbor relationships are protocols of interaction and are defined as types that can be associated with any actor. Thus, in a simulation of birds in flight, a bird’s near neighbors might be any bird or other object that the bird can see from its current position and heading. In an organizational simulation, an actor’s near neighbors might be only those actors who are connected by formal organizational ties, and neighbor operations might include sending and receiving messages.

In summary, actors have threads of control, have behavior, are “born” and can “die”, and have significant performance advantages over non-threaded approaches.

2.7 Interacting with a Simulation

A simulation needs mechanisms through which it can be controlled. Thus we need to be able to start the simulation, set up simulation parameters, change parameters while the simulation is proceeding, and observe output from the simulation. None of these functions are part of the simulation: they either send information to the simulation or retrieve it from the simulation. Easel recognizes two distinct roles: facilitators, which allow for global control, introduction of new examples, and control of parameter values; and observers, which extract values of the simulation parameters to do statistical analysis or drive graphical depictions.

Easel's visibility rules allow selected actors to play the role of facilitators or observers without any extra mechanisms.

2.8 An Example of a Simple Easel Simulation

As a gentle introduction to Easel, let's consider the following example. Ants are an interesting example of emergent algorithms, because an ant colony as a whole can evince behavior that is complex, but without global visibility and central control. As a contrived, entomologically unrealistic example, consider the question of how ants could form a circle without any communication among them. Here is an Easel simulation that shows how it might be done:

```
# We are simulating an ant hill
ant_hill: simulation type is                                 # 1
   v :: view := ?;
   ant_list :: list := new list any;

#  Life cycle of an ant
ant(id: int): actor type is                                  # 2

   # Ants have a position and a heading
   heading :: number := random(uniform, 0.0, 2.0*pi);         # 3
   x :: number := 250.0;
   y :: number := 250.0;

   # Create a depiction
   depict(sim.v,                                             # 4
      var offset_by(paint(circle(0.0, 0.0, 5.0),
      (firebrick)), var x, var y));

   # Pick a heading, walk out, walk back, repeat
   for every true do                                         # 5
      heading := random(uniform, 0.0, 2.0*pi);
      for h: each (1 .. 20) do
         x := x + 10.0 * cos heading;
         y := y + 10.0 * sin heading;
         wait 1.0;
      heading := heading + pi;                               # 6
      for h: each (1 .. 20) do
         x := x + 10.0 * cos heading;
         y := y + 10.0 * sin heading;
```

```
        wait 1.0;

#  Create the simulation and start its actors
ant_circle(n: int): action is                                      # 7

   # Create an ant hill and its view
   ant_sim :: ant_hill := new ant_hill;                            # 8
   ant_sim.v := new view(ant_sim,
      "Ant Circle", (papayawhip), nil);

   # Create the ants
   for i: each (1..n) do                                           # 9
      push(ant_sim.ant_list, new (ant_sim, ant i));

   # Wait for simulation to finish
   wait ant_sim;                                                   # 10

ant_circle 50;
```

We first declare a simulation type (ant_hill) at the line labeled 1. This simulation type has two attributes in addition to the predefined attributes of simulations: the view (v), which is used to portray the ants, and a list of ants (ant_list) which can be used to reference ants (for example, by iterating over the list).

At the line labeled 2, we declare the ant actor type. The actor is a predefined type in Easel that has the property of being threaded. This means that an actor is an independent process, has its own memory allocated (while it exists), and requires some CPU time to update its internal state and interact with other actors.

The properties of the ant type are defined next in line 3. These simple ants have only three properties: their orientation, an x coordinate, and a y coordinate. Note that the direction for any specific ant is defined randomly. Thus, each ant starts off at the center of the coordinate system with random initial orientation.

At line 4 we specify that each ant is to be depicted using a firebrick circle with a radius of 5 units, offset by whatever the ant’s current coordinates are. It is the call on var that ensures that the depiction of the ant is continuously updated in the display as the ant’s x and y coordinates change.

Starting at line 5, we provide the basic simulation loop through which the behavior of each ant is defined. This loop manages the thread of control for the ant in question and defines the ant’s behavior. In this simple case, the ant moves out 20 steps, then turns around and moves back to the center.

The procedure starting at line 7 (ant_circle) is the facilitator that manages the simulation. It first creates a simulation (ant_hill) and its view at line 8, then allocates the ants at line 9. Once a new ant has been created, it initiates a thread of execution that allows the ant to behave as specified by the ant's type (at lines 3-6).

Some miscellaneous observations of the program are worth making at this point. Note the type hierarchies: ant_hill is a subtype of the simulation type, while an ant is a subtype of the actor type. The code structure is defined through indenting and "outdenting." Comments extending to the end of the line are prefaced by the pound sign (#).

The ant example is interesting for a number of reasons. First, it demonstrates the concept of emergent properties. In this model, the emergent property is the circle that the ants generate as they move away from the nest. No individual ant has knowledge of the fact that it is part of this circle, yet, viewed globally, this is the shape that they collectively generate. Second, the example demonstrates the concept of neighbors. If neighborliness can be equated to nearness, then each ant has fewer neighbors as it moves farther from the nest until it finally has none. Third, other simulation languages that can address problems such as the ant circle often do so using a grid pattern that constrains each ant to be located in one of the grid's squares. This granularity issue has ramifications, for example, in accuracy of representation. While Easel could model the problem using a grid, no such grid need be imposed as each ant simply has position as one of its properties.

3 Synthetic Networks

In order to understand the general survivability characteristics of the Internet, we first focus on the construction of synthetic Internets that exhibit statistical properties comparable to the actual Internet. There has recently been considerable activity by researchers to gather data on the Internet (using, for example, Web “bots”). This data has provided insights about how to model synthetic variants of the Internet that, although different from the Internet in layout (routers and links), have statistical properties that reflect those of the Internet.

3.1 Existing Models

To date, there have been several attempts to develop synthetic models of the Internet. These include WAXMAN [Waxman 88], BRITE I and II [Medina 00], INET [Jin 00], and TIERS [Doar 96]. As with the approach taken here, these approaches do not attempt to generate exact replicas of the Internet’s layout, but to generate models whose statistical properties (for example, distribution of router size) reflect those of the Internet. The model developed in this report was motivated in part by the limitations of these earlier models (as identified by Yook [Yook 02]), but it was also developed to exercise the Easel simulation system in this area, to provide us with some first-hand experience in developing such models, and to support work in Internet survivability. For example, such models are important in examining the survivability properties of different Internet IP protocols.

Existing network generation models appear to be deficient in one or more aspects that are important to Internet modeling [Yook 02]. A significant omission is lack of appropriate spatial clustering models for the nodal population. These models place network nodes randomly, resulting in a fractal dimension D_f of 2.0 (see Section 3.3 for the definition of D_f). However, empirical evidence suggests that this clustering has a fractal dimension of about 1.5 [Yook 02]. Figure 3.1 illustrates the difference between two nodal distributions, one having a D_f of 2.0 and the other having a dimension of 1.41. Clearly there is a major difference. Clustering occurs because new nodes tend to co-locate in regions that already have a high density of nodes. This tendency is reflected in the way our model generates nodal distributions. The way this clustering occurs is central to the overall spatial connectivity of the network and will have significant implications for its survivability.

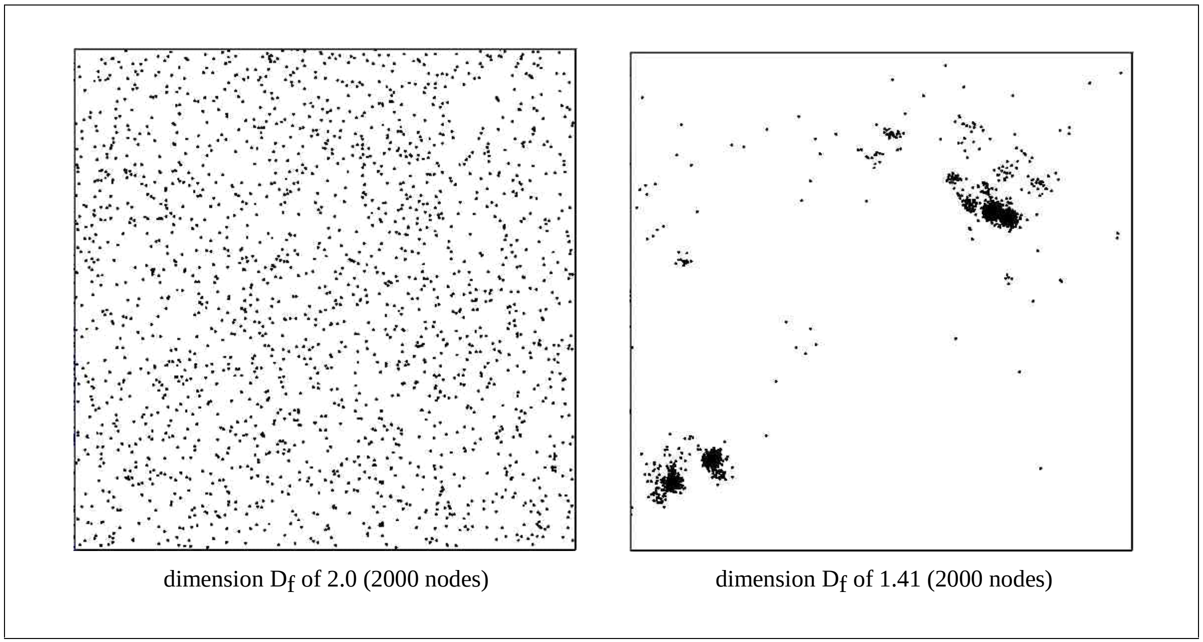


Figure 3.1: Nodal distributions with differing dimensions

Section 3 focuses on our approach to Internet topology modeling, how the approach is implemented in GENeSIS,[4] how GENeSIS is used, and finally provides examples of computations performed by GENeSIS. There are two appendices—one that reviews recent research in network topology and a second that describes how to use GENeSIS. This program can be accessed from the URL http://www.cert.org/easel/.

3.2 A Basis for Synthesizing Internet-Like Topologies

The approach we take to Internet topology modeling is based in part on the work of Barabasi and colleagues [Albert 02a, Albert 02b, Barabási 99, Barabási 02a, Barabási 02b, and Yook 02]. Underlying their approach to characterizing the Internet's topology are three assumptions:

1. The spatial distribution of the nodes[5] in the Internet forms a fractal set.
2. The network is built incrementally by adding nodes one at a time.
3. New nodes are linked to the existing network through a mechanism called preferential attachment (new nodes link preferentially to existing nodes that are more highly linked and spatially close).

Use of these simple rules results in networks that, for the most part, reflect the statistical properties of the actual Internet.[6] Each of these assumptions is reviewed below.

[4] GENeSIS is an acronym for Generation of Emergent Networks in Support of Internet Survivability
[5] Routers, hosts, and any other devices that are linked into the network are considered as "nodes."

3.3 The Internet's Spatial Distribution

After examining the spatial distribution of routers in North America and around the world, Yook determined that this distribution was characterized by a fractal set with dimension D_f ~1.5 [Yook 02]. This was determined using the "box counting" method [Chen 93], the method also used in the GENeSIS program to generate the spatial node distribution.

To determine the fractal dimension of a two-dimensional pattern, a square grid of various widths w is superimposed on an image of the pattern, in this case the distribution of network nodes. The number of grid boxes N(w) of width w that contain one or more nodes is then counted. The fractal dimension is defined through the equation $N(w) = w^{-D_f}$, which could also be stated as follows:

$$D_f = -\log(N(w))/\log(w) \qquad (1)$$

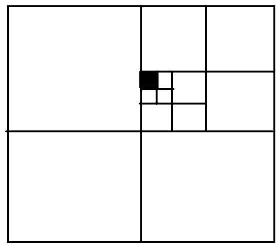


Figure 3.2: Basis for fractal box algorithm

In the GENeSIS program, the inverse procedure is used. Here we start with an empty square box, diving it into four quadrants. Each quadrant is recursively divided into four sub-quadrants and so on, down to a specified low level of granularity. Thus if we do four recursions we obtain 2^4*2^4 = 256 low-level boxes. To determine the box in which to place the next node, we select one of the top-level quadrants, biasing our selection preferentially to that quadrant that has the most number of existing nodes (if this is the first node placement then the selection will be purely random). Upon selecting one of the quadrants, we repeat the procedure using the subquadrants within this quadrant, and so on recursively down to the lowest-level box. At this point we place the node at a location determined by a normal distribution, centered in the middle of the box and with a standard deviation equal to half the box width. This procedure is repeated as each node is incrementally added to the population. Note two points. First, the degree of nodal clustering resulting from this procedure will depend on the strength of the bias used in quadrant selection. This bias can be changed through a fractal clustering exponent on the number of nodes in each of the boxes.[7] By varying this exponent, one can influence the value of the fractal dimension. Second, in order not to bias the clustering to any preferred box alignment, each time a new node calculation is performed, the origin of the box coordinates is randomly shifted.

[6] There is still some disagreement about the complete accuracy of the resulting networks, at least as far as these assumptions go with respect to modeling the Internet's autonomous systems [Chen 02].

[7] For example, an exponent of 2 will bias the selection to the square of the number of nodes in each box.

3.4 Incremental and Preferential Attachment

Once the spatial population of nodes has been defined, the nodes can then be linked. This is performed by incrementally connecting the nodes in the sequence they were generated (incremental attachment). As each node i is added to the network, it can be linked to one or more nodes j already in the network. To accomplish this, a selection function of the following form is used, where numLinksj is the number of links that currently attach to node j and $\text{distance}_{i,j}$ is the distance between nodes i and j:

$$sf_{ij} = (numLinks_j)\alpha \;/\; distance_{i,j}^{\;\sigma}, \qquad 0 <= \alpha <= 1 \qquad (2)$$

The actual node(s) j to which the node i is attached is probabilistically selected, based on the magnitude of the sf_{ij} values. With α=1, we get a topology that is strongly hub-based, and this is characterized by a power law in the distribution of numbers of links (see Scale Free Networks in Appendix A). With α=0, only distance is important in determining linkage, and the topology is more like a fishnet. As discussed in Survivability Implications in Appendix A, these differences have strong implications on network survivability. Empirically, it was found [Yook 02] that σ~1, but other relationships are possible, so this parameter provides that flexibility. For example, for σ =2 we would get an inverse square law on links and thus a greater weight would be given to nearby nodes.

Nodes need not be symmetrically connected. In other words, if node X connects to node Y, Y need not be connected to X. A connection probability p is specified such that if p = 1.0 then symmetric connectivity is guaranteed. If p = 0.0 then the reverse connections are not made.

3.5 Cliques

In addition to spatial clustering, clustering can take place through association between nodes. If node A is linked to node B and node B is linked to node C, in many cases there is a greater than random probability that node A is also linked to node C. This phenomenon is central to social networks—if Tom knows Mary and Mary knows Jean, then there is a significant probability (relative to random) that Tom will also know Jean. This probability has been estimated to be anywhere between a few percent to as high as fifty percent [Girvan 02, Newman 02], depending on the size of the network. If the probability were to reach 100 percent then the network would be fully connected (everyone would know everyone else), a situation that would only apply for very small networks. Clearly associations formed by cliques can have a significant influence on how viruses propagate through email lists, and that has motivated the incorporation of clique behavior into the model.

In GENeSIS, clique behavior is simply modeled by specifying that if node A associates with node B, and node B associates with node C then, with a certain probability greater than random, node A also associates with node C. This probability can vary anywhere between 0 to 1 (i.e., $0 <= C_f <= 1$). This model does not, however, account for subpopulations within a population where local “cliquing” may be higher than average.

3.6 The N-Squared Problem

Generating the network is an inherently n-squared problem, i.e., the n^{th} node that is incrementally inserted into the network searches all existing n-1 nodes in order to find the most desirable one(s) to link to. With very large n this is computationally very intensive. One simple approach to reducing the magnitude of this problem is to select and evaluate only a random sampling of all the existing nodes rather than every node. However, if too few random nodes are sampled, then the statistical properties of the network may be skewed. In particular, the node degree may no longer follow a power law. The following is a brief experiment into the effect of sampling size.

GENeSIS allows one to enter a sampling size. Thus, for example, if one generates a network of 1000 nodes one can specify a sampling size of 250. While the growing network is below 250 nodes, all nodes are selected; while between 251 and 1000 (say 532 nodes), a random selection 250 nodes out of the 532 nodes is made. Of course, if the sampling size is the same as the number of nodes than all nodes are evaluated.

The charts in Figure 3.3 show the results of 24 parametric runs involving 250 and 1000 nodes. In each case, there were 12 runs using different sample sizes. In the 250 node case, there were runs on samples of 62, 125, and 250 nodes; in the 1000-node case, there were runs on samples of 250, 500, and 1000 nodes.

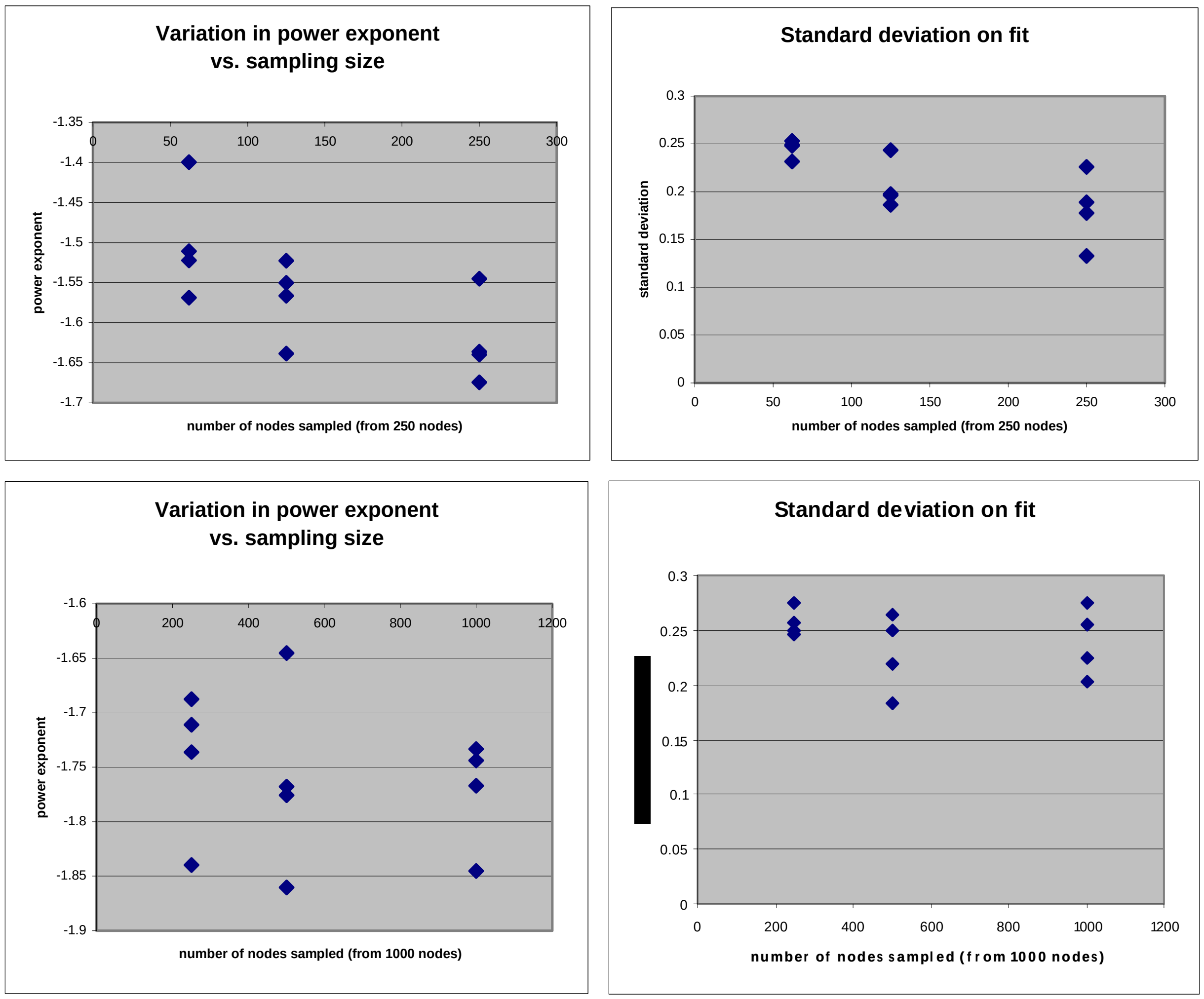


Figure 3.3: Results of parametric runs with different numbers of nodes

From the 250 node data it appears that, as the sampling size increases, the power law exponent somewhat decreases in size (becoming more negative), while the corresponding standard deviation on the linear fit to the data becomes smaller. However, it is difficult to see such trends in the 1000 node data (i.e., where confidence should be greater). Thus, at least within the limited range of these experiments, there does not appear to be any strong dependency of the power law exponent on sampling size.

3.7 A Description of the GENeSIS Program

GENeSIS generates networks. While it is written in the simulation language Easel, there are no time-dependent aspects to the computation. Hence it is not a simulation and does not use Easel's simulation features. An overview of functional elements in GENeSIS is shown in Figure 3.4. The two major components of the GENeSIS program are the fractal computation of the node spatial distribution and incremental network generation. One may run GENeSIS so that the network is generated immediately after the node population has been computed (option 1 in Figure 3.4), or one may capture the node population data in a separate file that can subsequently be used to generate a network (option 2 in Figure 3.4). The latter option allows one to generate different networks based on the same nodal distribution.

GENeSIS generates a variety of graphical outputs to allow assessment of the properties of the node/network properties to be made. Both the node and network spatial distributions can be graphically displayed. The fractal dimension of the nodal distribution is calculated using the box-counting method described in Section 3.3. *Log N(w)* is plotted against *log w*. In the network generation component, two analyses are made. First, the network's node degree[8] distribution is plotted and the associated power law exponent computed. Second, the distribution of link distances is plotted and the associated power law exponent is computed.

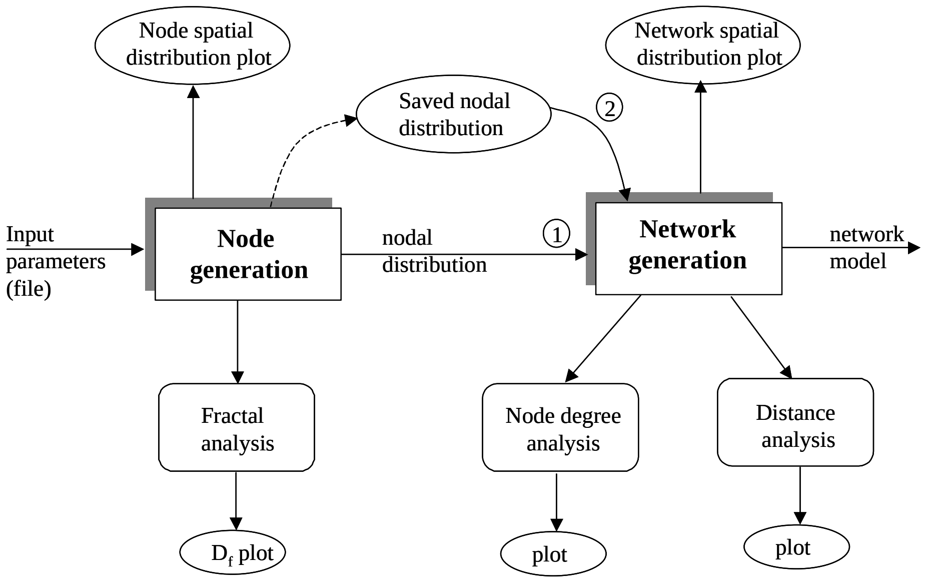


Figure 3.4: The structure of the GENeSIS program

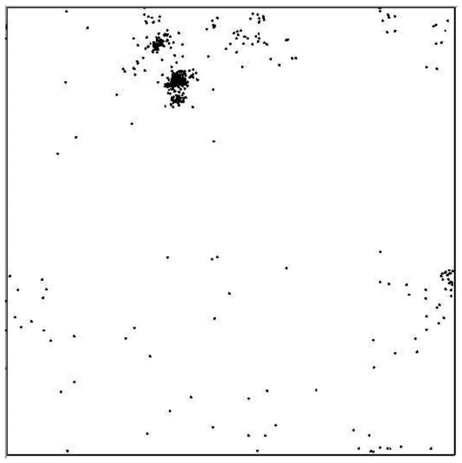
[8] The node degree of a node is the number of links associated with a node.

3.8 Network Synthesis Using GENeSIS

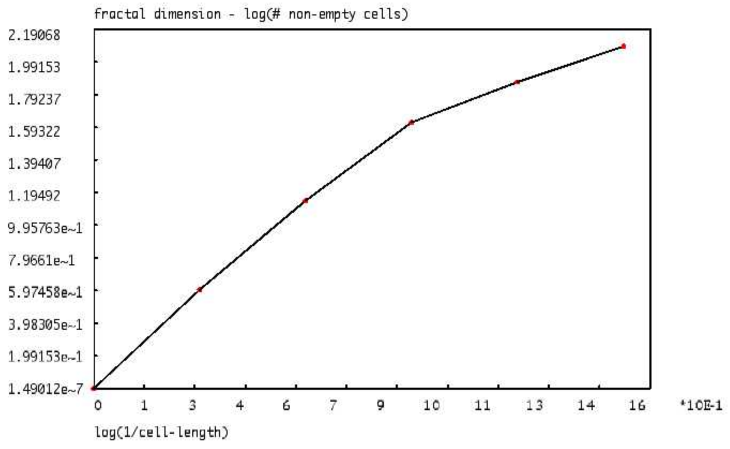
In this section we will first generate node distributions and look at their properties, and then based on these distributions we will investigate networks and look at their properties.

3.8.1 The Fractal Nature of Nodal Distributions

The following three figures show nodal distributions for 500, 1000, and 2000 nodes respectively and their associated fractal dimension characteristics. The 500 and 1000 node data sets were extracted as subsets of the 2000 node data. The fractal distributions show the -log (box size) against the log(number of boxes of that size that contain at least one node). The slope of this line is the fractal dimension D_f. These data were made at a resolution of $2^6 * 2^6 = 4096$ boxes for the fractal calculation and clearly indicate the fractal pattern of the data. The dimensionality of each distribution is also quite similar.

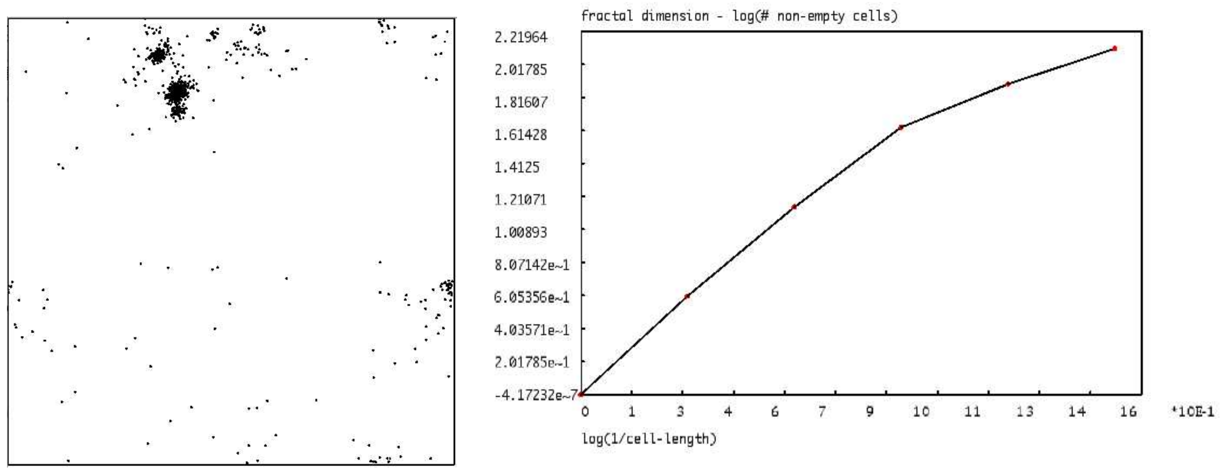


500 node distribution with clustering



Fractal dimension curve for 500 nodes

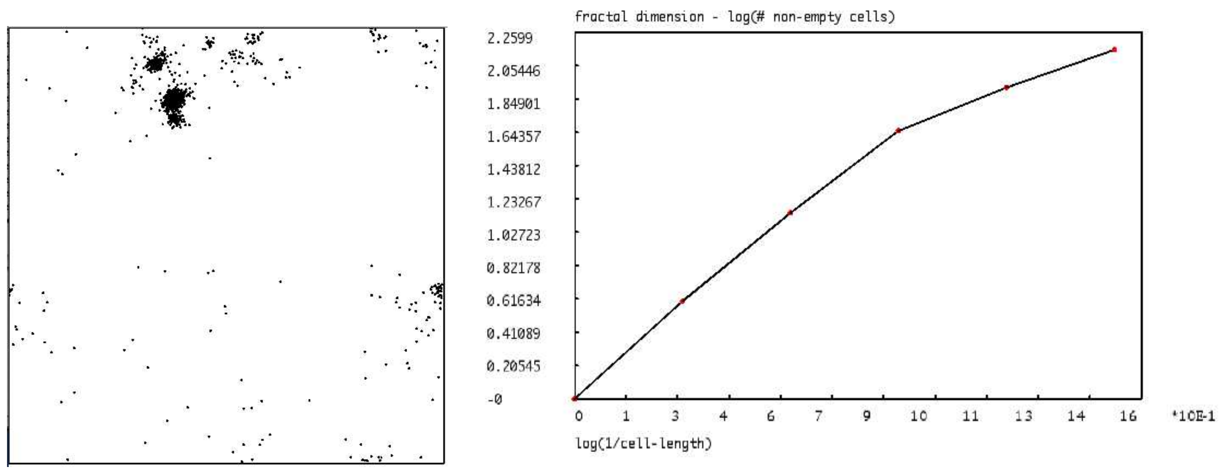
Figure 3.5: Distribution and fractal dimension curve for 500 nodes



1000 node distribution with clustering

Fractal dimension curve for 1000 nodes

Figure 3.6: Distribution and fractal dimension curve for 1000 nodes



2000 node distribution with clustering

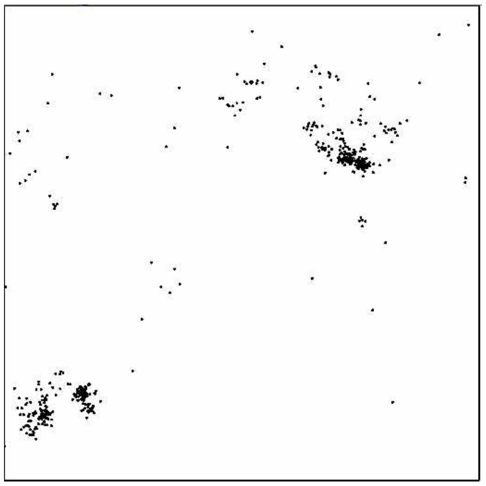
Fractal dimension curve for 2000 nodes

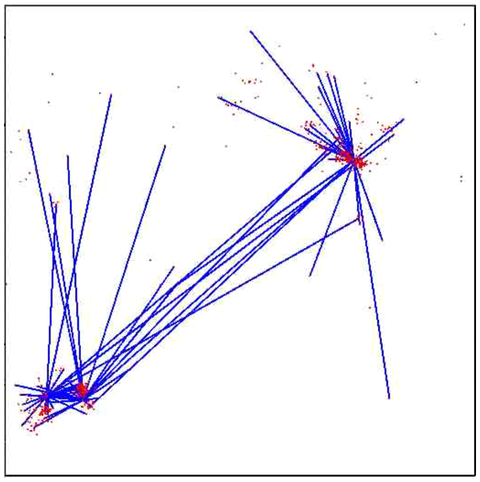
Figure 3.7: Distribution and fractal dimension curve for 2000 nodes

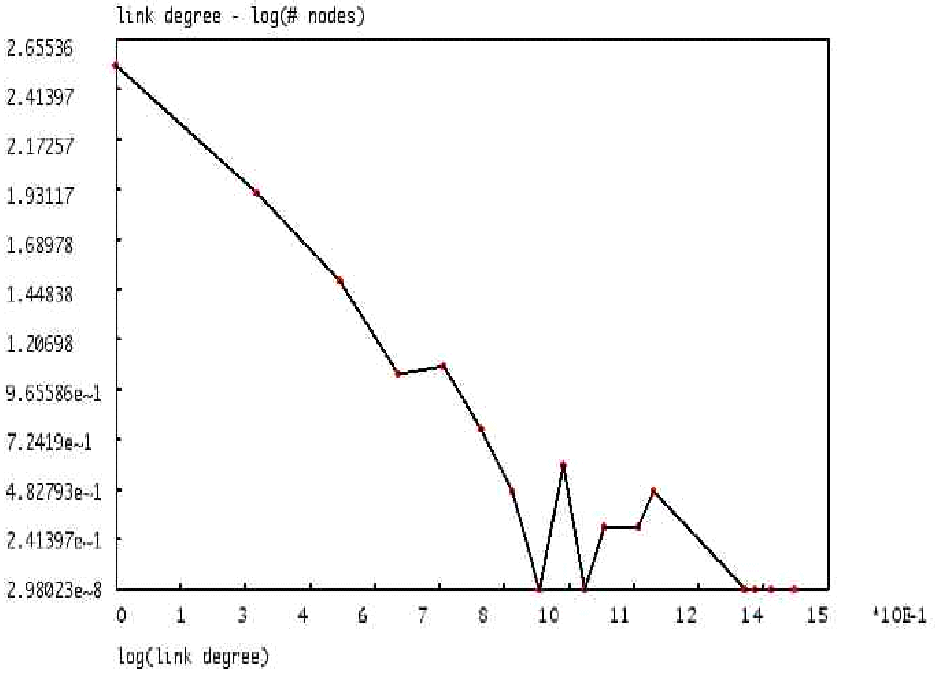
3.8.2 Variability of Distributions Based on 500 Nodes

This section illustrates the variability of nodal distributions and typical networks generated from these distributions. These runs were made for 500 nodes and at a resolution of $2^6 * 2^6 =$ 4096 boxes for the fractal calculation. These networks were generated such that a newly added node only links to one existing node. With respect to the network layouts shown in Figure 3.8 and Figure 3.9, a link is suppressed graphically if either end of the link is attached to a node having less than 25 links. This helps highlight the network structure—displaying all links can result in a very messy picture.

The linearity of the power law distributions is not very accurate due to the limited number of nodes (500) in the network.

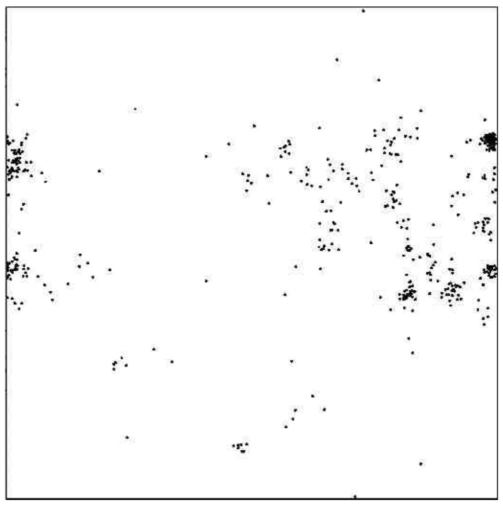

Fractal population with 500 nodes


Network with 500 nodes

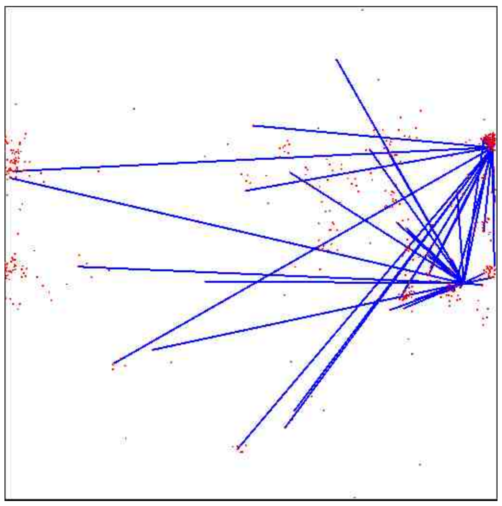


Link degree distribution (α = -1.575)

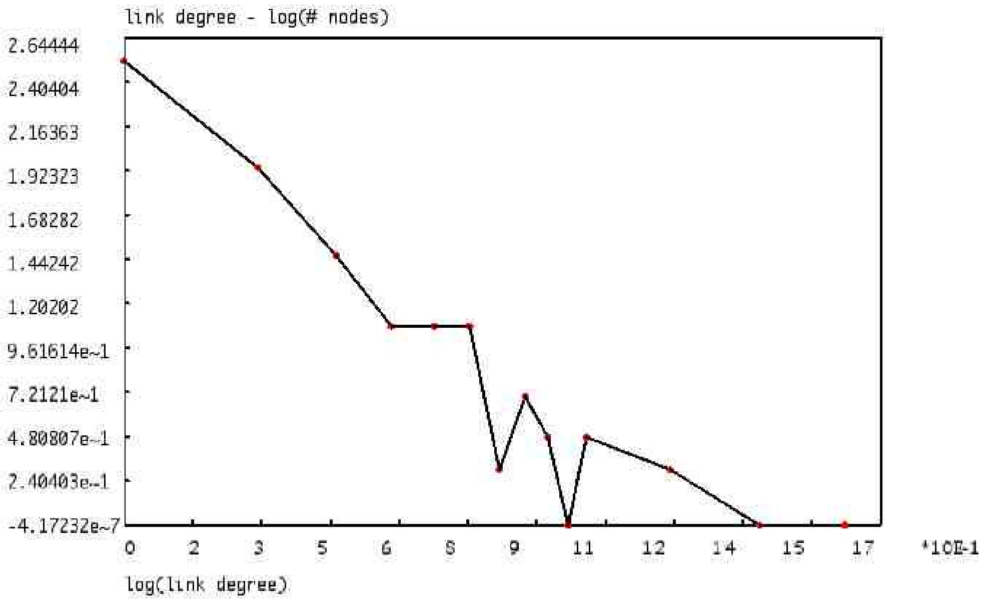
Figure 3.8: Case 1 showing variability of nodal distributions based on 500 nodes



Fractal population with 500 nodes



Network with 500 nodes



Link degree distribution (α = -1.644)

Figure 3.9: Case 2 showing variability of nodal distributions based on 500 nodes

3.8.3 Effect of Cliques

The degree to which individuals form cliques will have an effect on the linking structure of any network. This was examined by comparing two cases in which the nodal distribution was the same but the linking clique coefficient (see Section 3.5) was varied. In one case the clique coefficient $C_f = 0.0$ while in the other $C_f = 0.25$. Thus if A is linked to B and B is linked to C then there is a 25 percent probability that A is linked to C.

The two networks can be seen in Figure 3.10 and Figure 3.11. Note the much more dense linkages in the latter network, even though the nodal distributions in the two cases are the same.

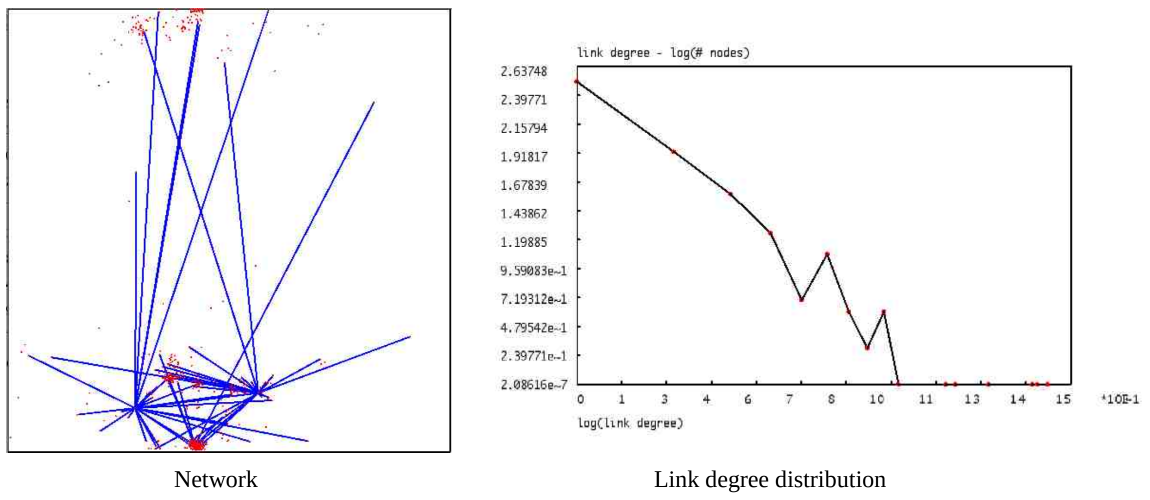


Figure 3.10: Case 1 (where $C_f = 0.0$)

It is interesting to observe that the network with a high C_f no longer obeys a power law in degree distribution (see Figure 3.11).

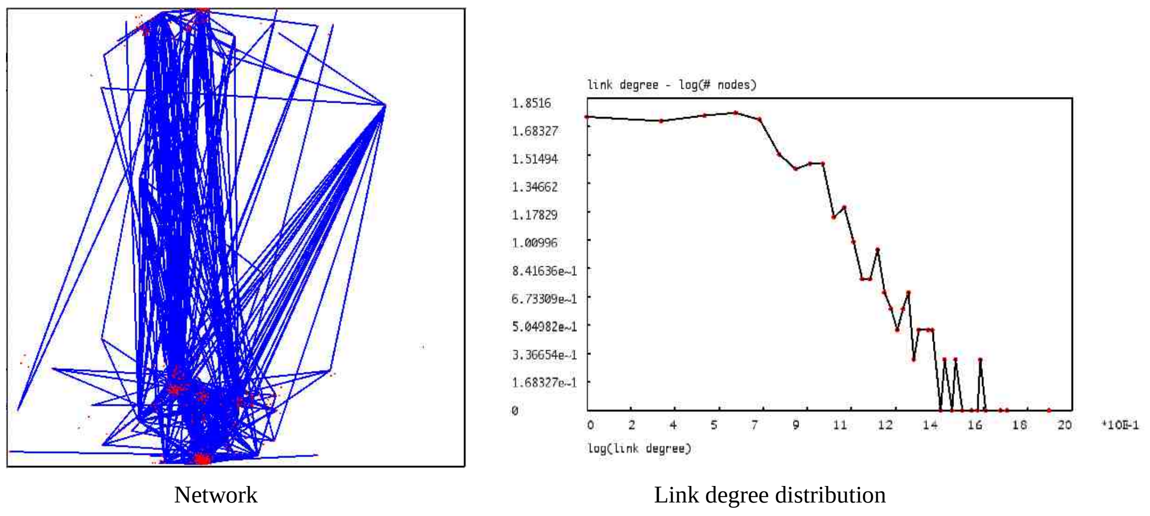


Figure 3.11: Case 2 (where $C_f = 0.25$)

3.8.4 Varying the Network's Topology

The topologies illustrated in Section 3.8.3 were generated using node degree and distance to determine linkage. The following networks show some variations on topology solely on the use of distance as a criterion for linkage. While these networks do not reflect the Internet's topology, they are interesting since, lacking large hubs, they may be more survivable [Albert 02b]. They may also better describe other networks such as the Interstate or railroad systems. Note that the nodal distributions for the random networks in Figure 3.12 and Figure 3.13 are the same.

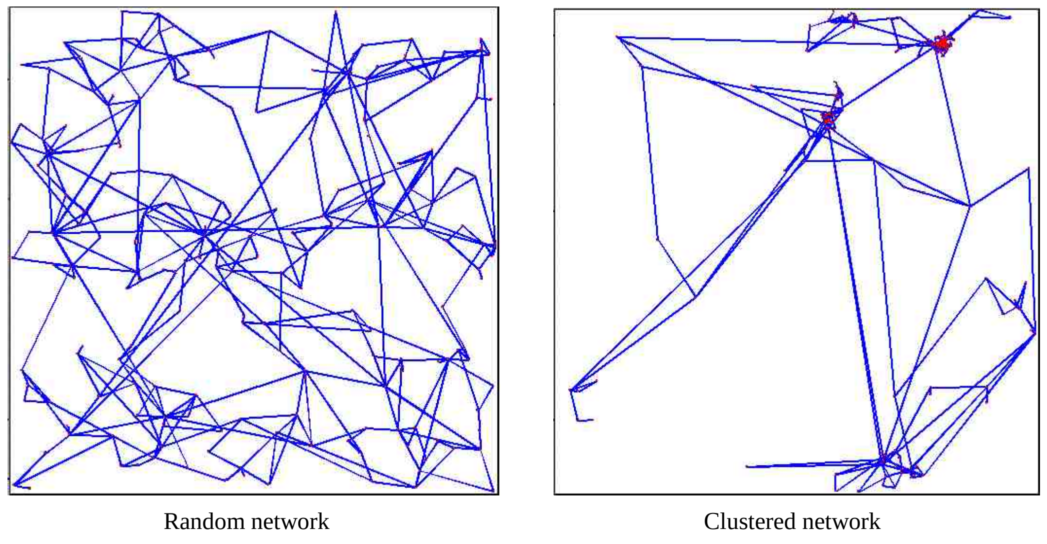


Figure 3.12: Variations in topology with distance as the only criterion for linkage

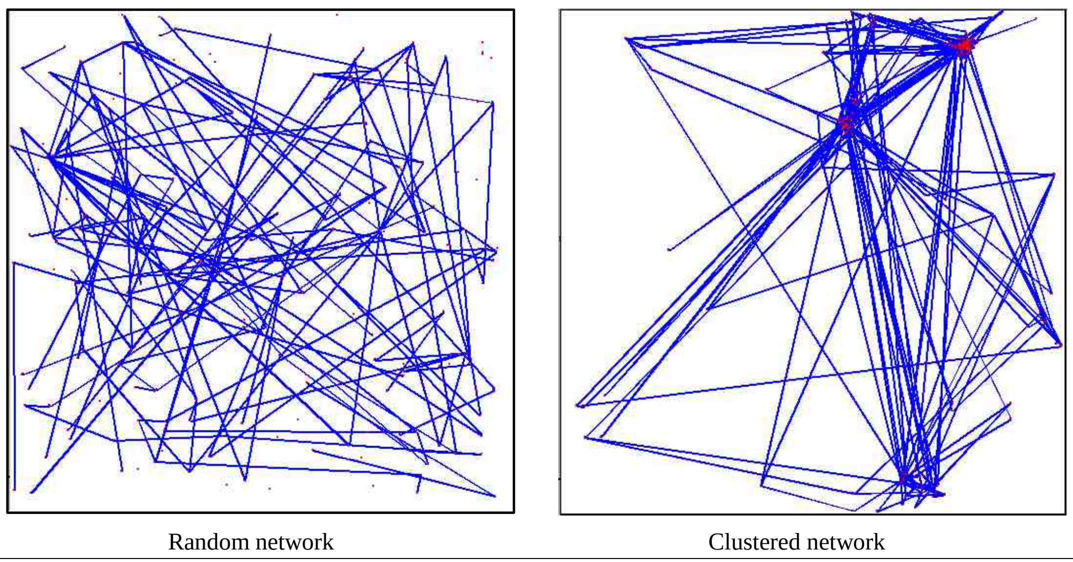


Figure 3.13: Variations in topology with distance and node degree as criteria for linkage

4 Simulating Virus Propagation with Easel

This section briefly explores the use of Easel in simulating computer virus propagation. The models are quite simple and are not intended to be rigorous. Rather, they illustrate the use of Easel in this area and in particular demonstrate how network topologies developed by GENeSIS can be imported into and used by a virus simulation. Extensions to the simulation models described here could easily be developed to make them more realistic.

Differences in network topology may have a decisive impact on the ability of viruses to propagate. The paper *Epidemic Spreading in Scale-Free Networks* [Pastor-Satorras 02b] and related work [Pastor-Satorras 01a, Pastor-Satorras 02c] are of particular interest. The authors claim that the very property that makes scale-free networks efficient (i.e., existence of large hubs) is also responsible for their propensity to propagate viruses efficiently. This property contrasts with the propagation of biological viruses that depend on social networks, where nodes (people) do not have the same high concentration of connections. Nor are people able to infect neighbors in the massively parallel way that computer viruses can infect connected nodes. The major claim of this research is that there is no critical threshold for virus spreading and that viruses can continue to remain in the network at low levels for an indefinite period of time (which has been observed). However, this work is based on somewhat idealized assumptions of, for example, an infinite population and the consequent lack of spatial modeling in infection propagation. In addition, the assumption that the Internet topology is the appropriate one for virus spreading may be in doubt. Many viruses spread through e-mail (buddy) lists, and this "network" topology may be quite different from that of the Internet [Patch 02]. It has not been shown that it obeys a power law, and it would be insightful to examine the distribution profile of "buddy-list" data. In a subsequent paper, "Immunization of Complex Networks [Pastor-Satorras 02a], the authors suggest that targeted immunization schemes that focus on nodal connectivity can sharply reduce vulnerability to epidemic attack (not an unreasonable conclusion). This claim is supported by the paper "Halting Viruses in Scale-Free Networks" by Dezsö and Barabási [Dezsö 02]. However, these papers deal with simplified analytic models (sometimes with verification through simplified simulation), and they lack critical real-world behaviors.

A recent paper by Staniford and colleagues [Staniford 02] presents some hypothetical models of worm propagation. The issues cited in this paper are of concern because the proposed mechanisms either result in (*a*) extremely rapid propagation throughout the Internet (in a matter of minutes) or (*b*) very stealthy propagation. The latter would allow large numbers of "zombie" machines to be set up in preparation for a potentially catastrophic attack. In all of these cases, an accurate understanding of the appropriate network topology is central to predicting how serious these attacks would be. Staniford's paper addresses this issue, but additional work in this area needs to be done.

The aim of this section is to illustrate the use of Easel in modeling propagation. We will look at a variety of issues, particularly two issues that were not fully explored by the above papers. First, we will look at clustering of network nodes as a mechanism for enhancing the survival of viruses. This issue was identified in the paper "Open Problems in Computer Virus Research" [White 98]. A second issue that will be explored is virus propagation through buddy lists. Quoting Jon Kleinberg, "The real network on which viruses spread is an invisible network of who talks to whom sitting on top of the Internet, and that's a network that we have less ability to measure at the moment" [Patch 02].

Individuals who use the Internet tend to form communities whose frequency of communication is high (for example, employees of a business or members of a social club). Individuals in these groups may be more prone to being reinfected through other members. Members may then spread the virus outside the group at a lower frequency. In other words, pockets of the virus may remain active within close-knit communities, while the prevalence of the virus is low or non-existent in the general population (see Section 3.5).

4.1 The Easel Virus Propagation Models

Two standard models of infection are often considered. In the first, reinfection of an individual can occur after the infection has been eliminated. This is called the susceptible-infected-susceptible (SIS) model. In the second, immunization or death prevents reoccurrence of the infection. This is called the susceptible-infected-removed (SIR) model. Both of these are examined below.

The model accounts for the delay incurred in developing an anti-virus signature and the probability that a host's user has installed the software that supports the signature. If the host is in the susceptible state, then to become infected (*a*) the host must receive the virus from another host, and (*b*) the virus signature must be unavailable or the anti-virus patch must not have been installed. If the host is infected, then the virus exploits the list of email contacts to propagate the virus to other hosts. In the SIS model, the host is then cleared of the virus, but returns to the susceptible state. In the SIR model, the host is permanently vaccinated against this particular virus. This is summarized in Figure 4.1 below. Note that these models use arbitrary time units. While the time units are thought to be relatively consistent, further work is required to pin down validated real-world values.

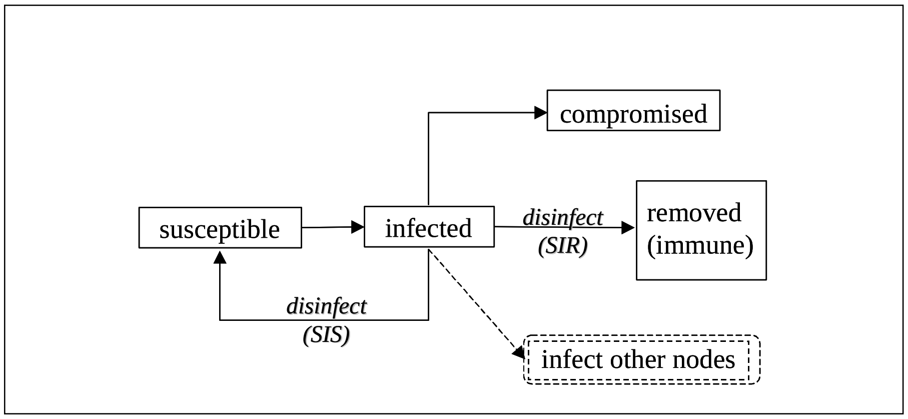


Figure 4.1: State model of virus propagation

The key lines from the Easel model are as follows:

```
# chance in each cycle of acquiring virus SW, if virus signature is available
if (gen.signature & gen.acquire>random(uniform, 0.0, 1.0)) then
    haveVirusSW := true;

if state = susceptible then
# in susceptible state
    clr := green;
    if virusReceive then                  # virus is received ...
        if (!gen.signature |              # but the virus signature not available ...
            # or virus signature is available but dont have antivirus S/W
            (gen.signature & !haveVirusSW)) then
            # then some of these individuals activate the virus
            if gen.activate>random(uniform, 0.0, 1.0) then
                virusReceive := false;
                wait random(exponential, 1.0/dt_act); # delay until virus file is activated
                newState := infected;
                compromised := true;
    wait gen.dt;
# in infected state
else if state = infected then
    clr := red;
    sendVirus(nodeList);
    if (gen.signature & haveVirusSW) then  # virus signature is available and have virus software
            wait random(exponential, 1.0/dt_fix); # delay until virus is eliminated
            if gen.modelType = SIS then           # for the SIS model
                newState := susceptible;
                else if gen.modelType = SIR then
                    newState := removed;
        else
                wait gen.dt;

# in removed state (SIR model only)
else if state = removed then
    clr := black;
    wait dt_removed;
```

While the behavior of each host is simple, the emergent behavior of the overall network is quite complex and sensitive to the network's topology.

The networks used in most of these virus simulations are shown in Section 3.8.2. Figure 4.2 illustrates typical transient plots from the virus propagation program using the first of these networks. These are for the SIS and SIR models respectively. Note that the SIR model results in nodes becoming immune while the SIS model does not.

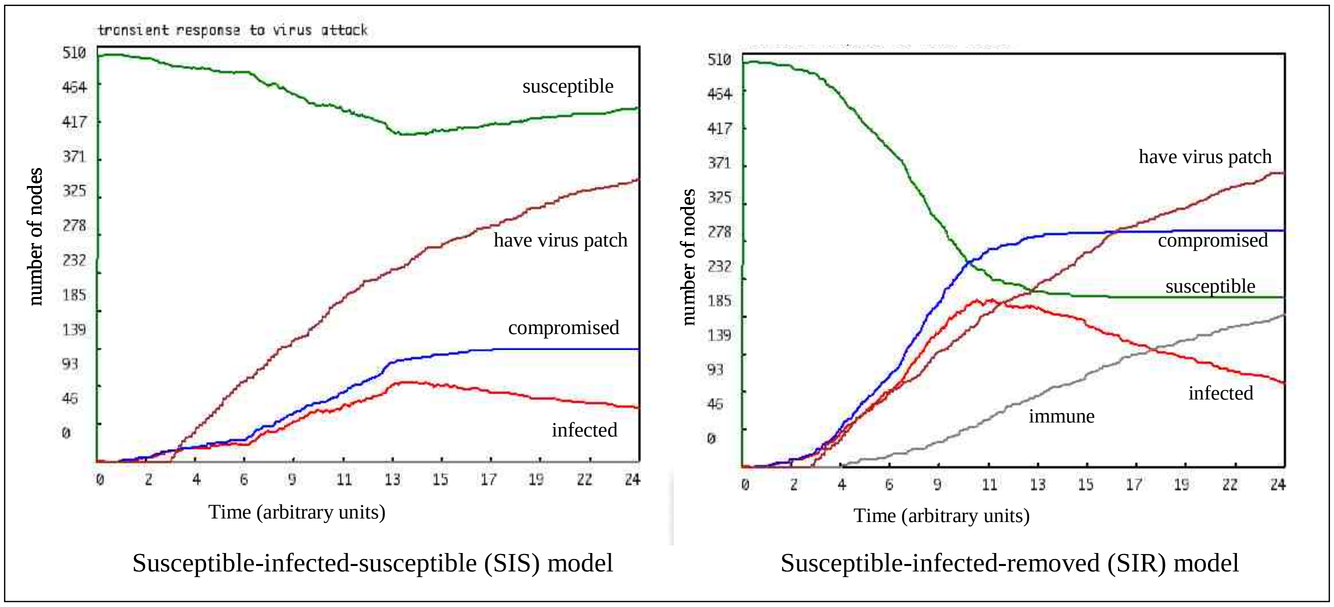


Figure 4.2: Typical transient plots from the virus propagation program using networks described in Section 3.8.3

4.2 Simulating Virus Propagation Through Networks

We now briefly look at the response to some parametric variations of attributes in the model. In all cases we use the network topology generation program GENeSIS to synthesize the networks and then import them into the virus propagation program. To examine these sensitivities, a base case virus propagation simulation was run. The simulation had the following properties:

- delay in developing anti-virus patch is 2 arbitrary time units
- 100 percent of individuals who receive virus activate it
- network is based on the topology of Figure 3.8
- SIR virus propagation model is used
- virus activation and system fix times are constant across all nodes
- node 0 was selected (this node has four immediate links) for initial infection
- cliquing probability is set to zero

The resulting transient response is shown in Figure 4.3. The stepwise characteristics result from the fact that the time for each node to activate the virus is the same. Thus it takes a fixed time for the infection to spread from the first victim to its immediate neighbors (for example, set X) and the same fixed time to spread from set X to its immediate neighbors (set Y), and so on. With stochastic variations in the delay times, these discontinuities become smoothed out (as in Figure 4.2).

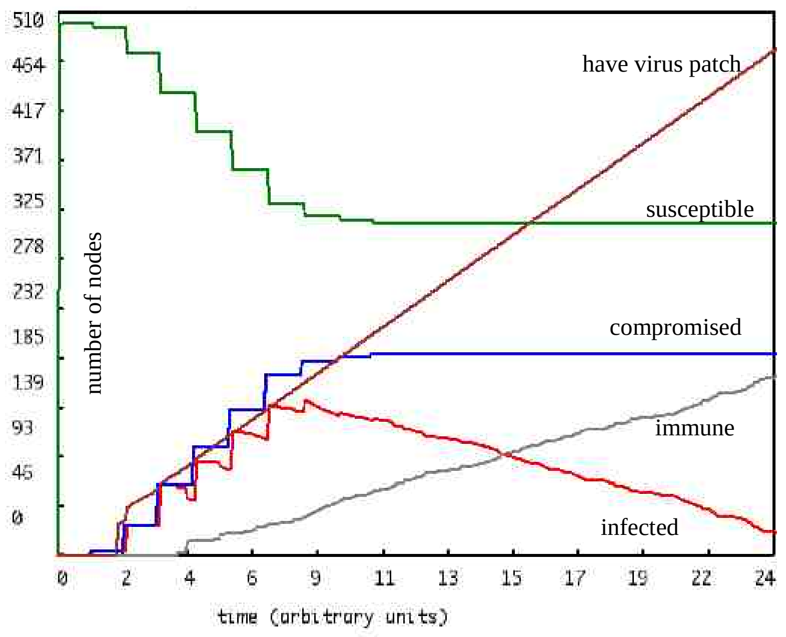


Figure 4.3: Baseline deterministic simulation of virus propagation (SIR model)

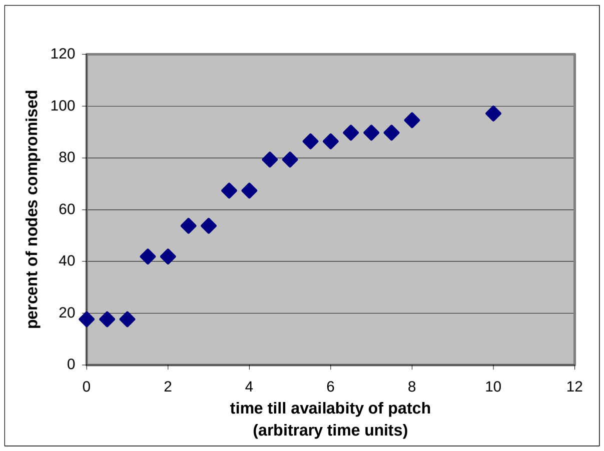
Table 4.1 summarizes the parametric runs based on the above simulation.

Table 4.1   Results of parametric variations on virus propagation attributes

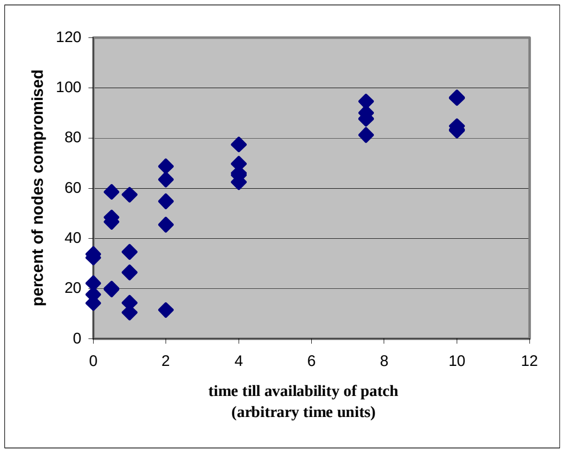
Case	**Description of run**
0	Base case
1	Effect of delay in virus patch availability
2	Effect of cliquing
3	Probability of activating a received virus
4	Topology modeled with distance-based incremental attachment only
5	SIS model of virus propagation
6	Stochastic variation in activation time
7	Stochastic variation in fix time
8	Node selected for virus insertion

Case 1: Effect of Delay in Virus Patch Availability

Case 1 examined the effect of delaying the availability of a virus patch.The rapidity with which anti-virus software vendors can release a virus patch is clearly important. However, what effect does this speed of response have on the spread of viruses? Figure 4.4.A shows the sensitivity to nodal compromise that results from different delay times in patch availability. These results are based on a constant time (T_{av}) for all individuals to install (activate) the patch once it has been released. The stepwise increment in the number of compromised nodes is again a reflection of the constant patch activation time (i.e., each node delays patch installation by the same amount after it has received the patch. Figure 4.4.B shows a more realistic response when stochastic variations in delay time are introduced. These variations are based on an exponential distribution having a characteristic time of $T_{av.}$ Stochastic variations of activation time appear to result in wide variations in the degree to which the population becomes compromised, particularly when the patch is rapidly released. Note that the model does not account for individuals who take measures to protect themselves before or after a patch becomes available.



A. Constant activation time



B. Randomly distributed activation time

Figure 4.4: Sensitivity of compromise to patch time

Case 2: Effect of Cliquing

Case 2 examined the effect of cliquing. The formation of cliques, that is, groups of nodes that have a high frequency of interaction, will have the effect of perpetuating a virus within these clusters of nodes. To assess the effect of clique formation, two sets of parametric runs were performed. The first set of six runs was based on a network in which no cliquing was modeled, while the second set of six runs was based on a network in which a cliquing probability of 0.15 was assumed.[9] These networks are shown in Figure 4.5, where it can be seen that the linking in the latter case is more local. (For clarity, links that connect to nodes with less than ten links at either end are suppressed). While these networks were based on the same nodal distribution, their linkage patterns were different, but the average link degree is the same in both cases. Because of the topological differences, comparing their responses to virus propagation requires statistical examination. For both cases, the last six nodes in the network (nodes 494 through 499) were initially infected. Although these nodes are not comparable with respect to their linkages, they are outliers in that they were the last to connect to the network. Six runs in each category is probably not statistically sufficient, but the results show some consistency.

[9] The cliquing model defines a probability p that if node A is linked to node B, and node B is linked to node C, then C is linked to A.

The cases where cliquing occurs appear to result in lower overall compromise, which can be explained by the fact that cliquing tends to result in islands of infection that are somewhat isolated.

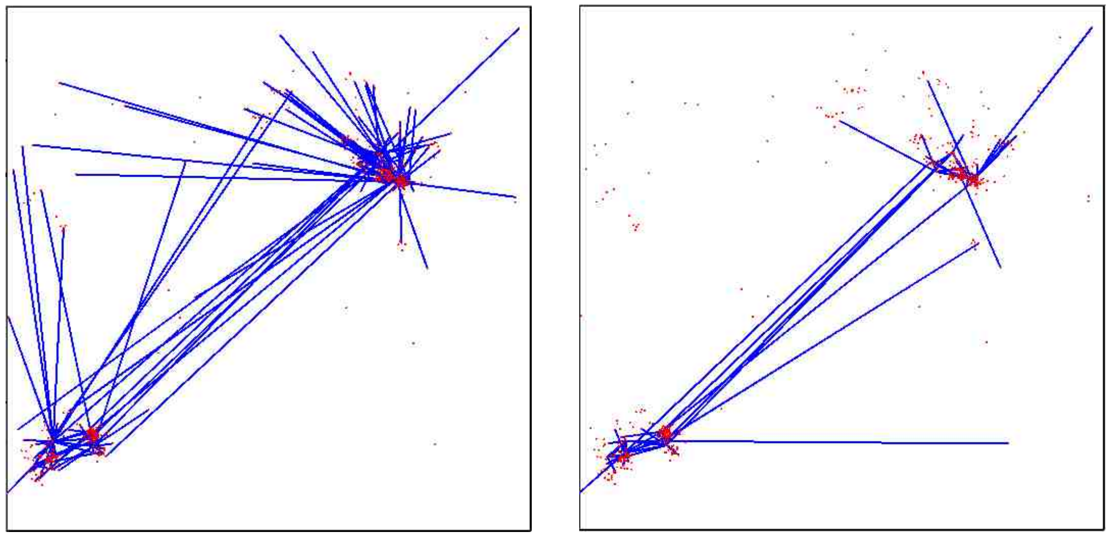

Network with no cliquing (cliquing probability = 0.0) Network with cliquing (cliquing probability = 0.15)

Figure 4.5: Effects of cliquing on virus propagation

The numbers in Table 4.2 reflect the final percentages of nodes that were compromised.

Table 4.2 Sensitivity of nodal compromise to cliquing

Network with no cliquing (cliquing probability = 0.0)						
Run #	**1**	**2**	**3**	**4**	**5**	**6**
	18.8	6.4	16.2	13.4	4.4	16.2
Network with cliquing (cliquing probability = 0.15)						
Run #	**1**	**2**	**3**	**4**	**5**	**6**
	2.2	4.6	0.8	3.0	1.0	18.8

Case 3: Probability That a Virus Is Activated

In case 3, the probability that a virus is activated is reduced from 100 percent to 50 percent. This reduction in activation results in a dramatic reduction in the number of compromised nodes—the two cases indicate compromise of only 1.4 and 0.8 percent respectively. A significant reason for this large drop can be attributed to the slower spread of the virus, which allowed greater application of the anti-virus patch prior to major build-up of serious infection.

Case 4: Topology Modeled with Distance-Based Incremental Attachment Only

In case 4, the topology of the Internet is strongly influenced by the presence of large hubs. As has been pointed out elsewhere [Albert 02b], this is both a strength (in terms of the ability to function despite a large number of random failures) and a weakness (in terms of the breakdown in connectivity when the large hubs are targets of attack). The following comparison illustrates the sensitivity of virus spread as a function of nodal topology.

In the two cases, the same 500-node spatial distribution is used. The first arrangement (the base case) generates links using both node degree and node distance criteria; the second only uses the distance criterion for link generation.[10] (These cases use the equation found in Section 3.4 with the values α=0.0, σ=1.0.) These networks are shown below in Figure 4.6. Although it appears that the former topology has many more links, the number of links for the two cases is virtually the same. Thus the average link degree is the same.

The distance-based topology indicates an increase in total number of compromised nodes (from 37.8% to 51.2%). This may result from the fact that, although the base case has more highly connected nodes, it also has more nodes that have few connections (despite the impression given by the figures). These more isolated nodes are less likely to become infected and may suppress the total number of compromised nodes.

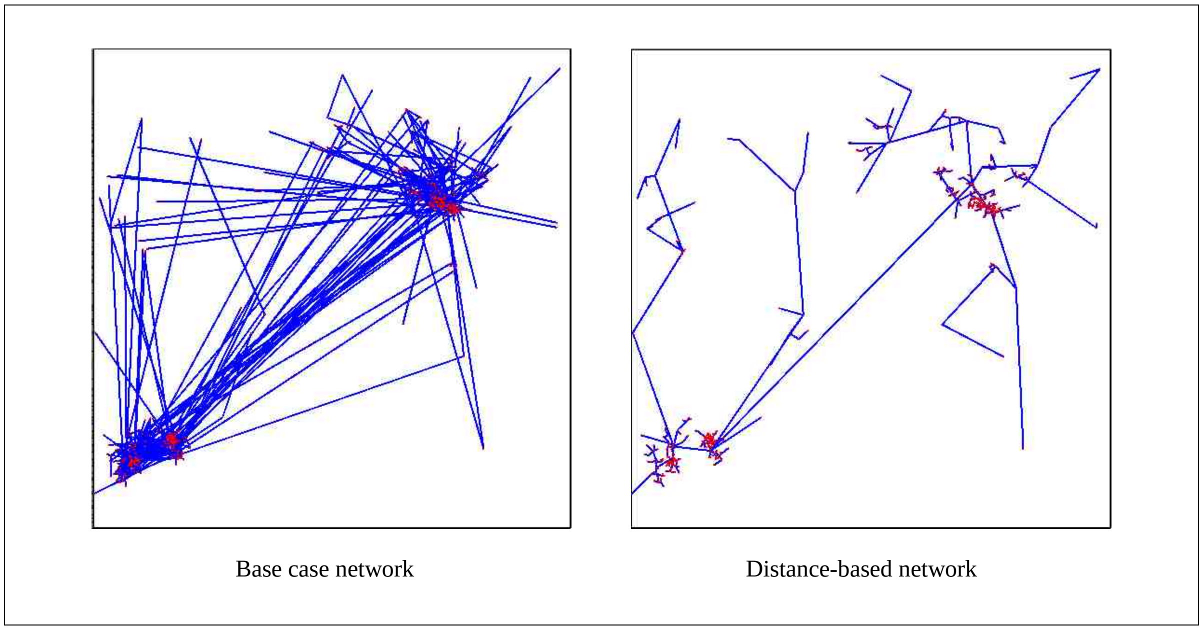


Figure 4.6: Networks showing the sensitivity of virus spread as a function of nodal topology

[10] The standard distance criterion in GENeSIS is probabilistic in the sense that the smaller the distance between two nodes, the more likely they are to be linked. However, the distance criterion used here deterministically links the closest node to the one in question.

Case 5: SIS Model of Virus Propagation

In case 5, the simulation was run with the SIS model. The number of compromised nodes for this model is the same as that for the SIR model. At first this may seem counterintuitive, as compromised nodes in the SIR model are removed from further involvement in virus spreading. However, even though the nodes in the SIS model return to the susceptible state, they have already contaminated 100 percent of their linked neighbors. Further contamination of these neighbors does not result in increasing the total pool of compromised nodes.

Case 6: Stochastic Variation in Activation Time

Case 6 examined stochastic variation in victim activation time. In this case, stochastic variations refers to the time it takes for victims to activate the virus once they have received it. These variations have the effect of smoothing out the responses (over those observed in Figure 4.4.). The overall effect on compromise is mixed—sometimes the result is lower and sometimes it is higher.

Case 7: Stochastic Variation in Fix Time

Case 7 examined variation in victim activation time. Stochastic variations of this parameter have no effect on level of compromise.

Case 8: Node Selected for Virus Insertion

Case 8 examined the initial fan-out of the virus. The ability of the virus to gain a foothold is critically dependent on whether it can establish itself early on in the network. Thus we examine the effect of the degree of fan-out from nodes neighboring the node that is first infected. Figure 4.7 shows the sensitivity of the spread of the virus to this fan-out for three cases, which differ only in the node that was initially infected. This clearly illustrates the property that initial fan-out of the virus (as reflected in the initial gradients of the curves) is important in predicting the severity of the viral outbreak.

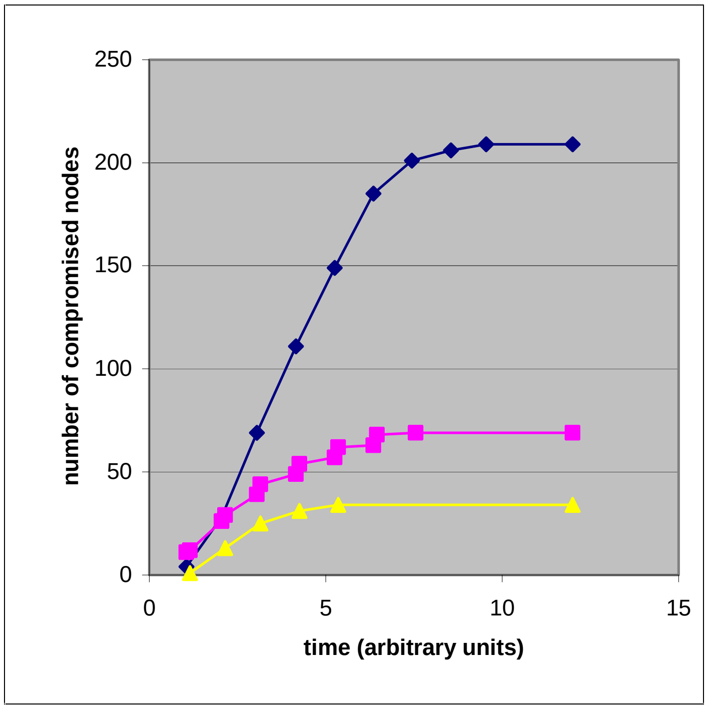


Figure 4.7: Effect of initial fan-out on total number of compromised nodes

4.3 Conclusions

There are several tentative implications that can be drawn from the simulations in this section.[11]

- The delay in patch availability has a noticeable effect on the ability of the virus to spread. However, if the patch is released with reasonable promptness, then the degree of compromise shows such wide swings (as a result, for example, of which node is initially infected) that some latitude in release delay may be acceptable.
- Cliquing has a tendency to isolate clusters of nodes, which can hinder viral spread through the whole population.
- When the virus activation rate (i.e., the fraction of nodes that activate a virus upon receiving it) is reduced from 100 percent to 50 percent, there is a dramatic drop in the compromise to the entire population.
- The link degree of the initially compromised nodes has a strong effect on the subsequent ability of the virus to compromise the entire population.

[11] Further work should be performed to collect empirical data upon which to base the numerical values used in this analysis and validate the responses of the simulations.

Appendix A: Survivability and Network Topology: A Survey of Recent Research

Introduction

The ability of the Internet to survive broad attacks, accidents, or failures is quite dependent on its topological characteristics, i.e., the properties that govern how routers and connected platforms are linked together. In the past handful of years there has been a significant paradigm shift in our understanding of how real-world networks are constructed. This is having a huge impact on modeling, not only of the Internet's topology [Barabási 02b, Chen 02, Magoni 02, Medina 00, Pastor-Satorras 01a, Vazquez 02, Yook 02,], but also of many other natural and artificial systems such as electrical grids [Stubna 02, Watts 98], social interactions [Barabási 99, Ball 02], and food webs [Montoya 02, Strogatz 01]. The increased understanding of how these systems are composed can shed light on how to make the Internet more survivable. The issues raised are new and different from those we commonly encounter in computer science. Quoting Barabási, "Increasingly we are realizing that our lack of understanding of the Internet and the Web is not a computer science question. Rather it is rooted in the absence of a scientific framework to characterize the topology of the network behind it." [Barabási 02b].

This appendix is intended to summarize what has been done in the field and provides extensive references to relevant articles for those who wish to dig deeper. It also suggests some directions we might take to address some of the survivability concerns.

Summary of Network Concepts

Pioneering work on examining network characteristics was done by Erdös and Rényi [Erdös 60]. They primarily focused on unbounded networks in which nodes were randomly connected. Such graphs are appealing since they can be investigated analytically and provide insights into, for example, the connectedness of subgraphs within the overall graph. In these models, any two nodes are connected with a probability p. Thus, for a graph with N nodes the total expected number of links in the network is pN(N-1)/2. In addition, Erdös and Rényi discovered that for large numbers of nodes, the distribution of the number of links attached to each node in the graph turns out to be a Poisson distribution [Erdös 60]. This is important since the Poisson assumption was often made in constructing synthetic networks. However, for many network topologies, including the Internet, the Poisson assumption turns out to be very poor. Another important factor that was examined is the so-called "diameter" of the network [Barabási 99]. The diameter is the average number of hops it takes to get from one node to another. In this regard, random networks have characteristics that differ significantly from networks such as the

Internet. While both types of network show a logarithmic increase in number of hops as a function of the number of nodes, the Internet needs fewer hops to get from one node to another [Albert 02a].

At the other end of the network topology spectrum are grid models. These networks conform to a regular lattice of links that often form a rectangular lattice.[12] They have high diameter, since it takes many hops to travel from one node to another.

Between the extremes of random and lattice networks is the most interesting class of networks—small world networks—of which the Internet is an example. Figure A.1 illustrates a "ring" model created by Watts and Strogatz [Albert 02a, Strogatz 01] and provides a context to demonstrate the evolution of a network from a regular lattice to random network. With probability *p*, one end of each link can be reconnected to another node. Two properties are of particular importance here: diameter and coupling. The former was discussed above, while the latter is a measure of how clumped the nodes are. If we choose a node *i* and select all its k_i immediate neighbors, then these nodes can have a maximum number of interconnections $k_i (k_i-1)/2$. If the actual number of interconnections is E_i, then the node is said to have a clustering coefficient of $C_i = 2E_i/k_i (k_i-1)$. The average of this over all nodes (C) is the clustering value of the whole network. Clustering is "good" since it means that immediate neighbors can always get to each other quickly—but it doesn't necessarily mean that there are short paths to distant nodes. As shown in Figure A.2, the clustering of the lattice model is high. However, because the normalized diameter, L(p)/L(0), is also high, it takes many hops to get to distant nodes. In the random network, the opposite is the case—local nodes may be many hops away, while distant nodes may have few hops. The intermediate small world graph (which has a small number of reconnections) is interesting since it has the best of both worlds—high clustering and low diameter.

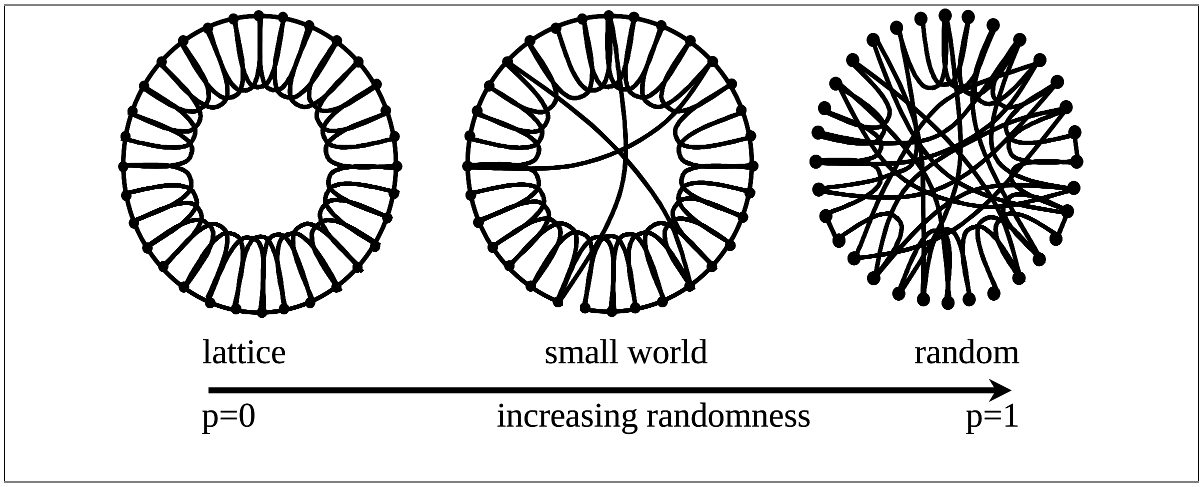


Figure A.1:Spectrum of simple network models

[12] See Figure A.1 for another lattice example.

Thus, we can get to neighbors quickly without sacrificing the need for short hops to distant nodes. This is a characteristic of the Internet. Note, from Figure A.2, that with only one percent of the links being reconnected we achieve the small world property. This has significant implications for the "survivability" of this simple model, since breaking these few crucial long-distance links rapidly increases the network's diameter.

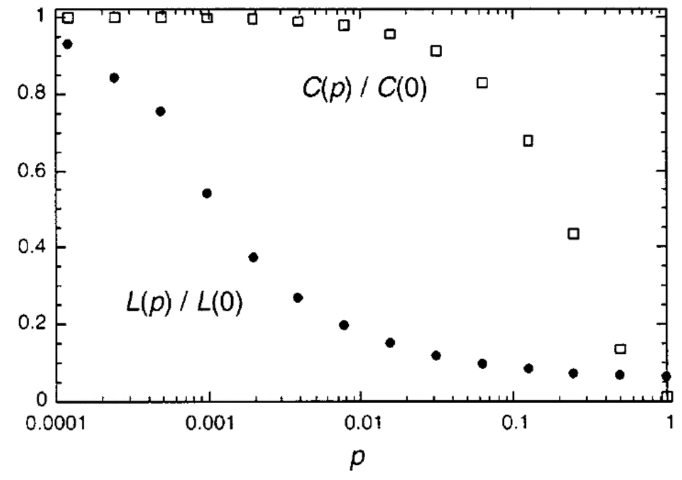


Figure A.2: Normalized coupling and diameter as functions of probability of links being randomly reconnected [Watts 98]

Percolation Theory

The Internet is designed with redundancy in mind—i.e., there are multiple paths through intermediate routers between source and destination nodes. However, if major routers are corrupted or destroyed, quality of service will be degraded—and the effective diameter increases. At some threshold point of destruction the network may become disjoint with non-communicating subnets (and the diameter between the subnets becomes infinite). This problem falls within the domain of percolation theory [Stauffer 94], which deals with the ability of a system to function under increasingly degraded conditions. One classical percolation model addresses the manner in which a fire consumes a forest [Malamud 02]. Below a certain density of trees, a spark may ignite some trees locally but fail to propagate. However, above a critical density,[13] the forest may be totally consumed. This is illustrated in Figure A.3, which shows that close to the critical density (0.593) the ability of the fire to propagate changes rapidly. Not only is percolation theory relevant to the ability of degraded networks to function, but it is also very relevant to epidemics, including the spread of computer viruses. Some viruses fail to propagate, while others take off with strong virulence. Understanding why is important and percolation theory should shed light on the issue. This simplified forest fire model also demonstrates an issue that is important to network survivability: cascading failure.

[13] This density turns out to be a very precise threshold value, 0.592, for a normalized maximum density of 1.0 (i.e., when the forest has no empty spaces).

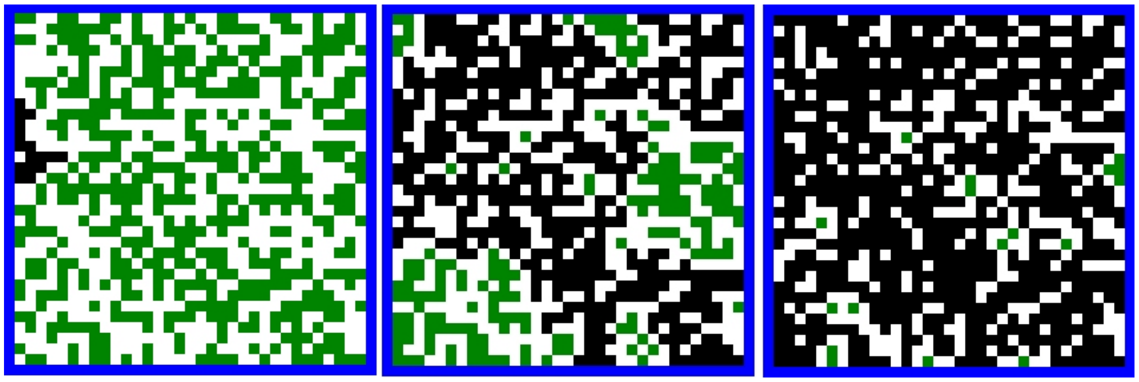


Figure A.3: Simulated forest fire with different tree densities (p) (Green represents unburned trees, black represents burned trees)

Scale-Free Networks

One area that has received wide attention recently is the statistical characterization of real networks. Because the Internet was a ready source of data, it was the first target [Faloutsos 99], but this rapidly broadened out to an examination of other fields [Albert 02a, Amaral 02]. This work has shown that many topological features of the Internet can be described through a power law of the form $P(k) \sim k^{-\gamma}$, an equation which states that the probability P of a node having k links is proportional to k^{γ} where γ is a constant somewhere between 2.0 and 3.0. The resulting topological difference between this class of network and random networks can be seen in Figure A.4[14]. These two distributions have the same number of nodes and links but their connectivities are clearly quite different. These characteristics are reflected in the shape of the link distributions (see Figure A.5). Because the power-law based topology does not have a characteristic peaked mean value (as in the Poisson case) it is often called “scale-free.”

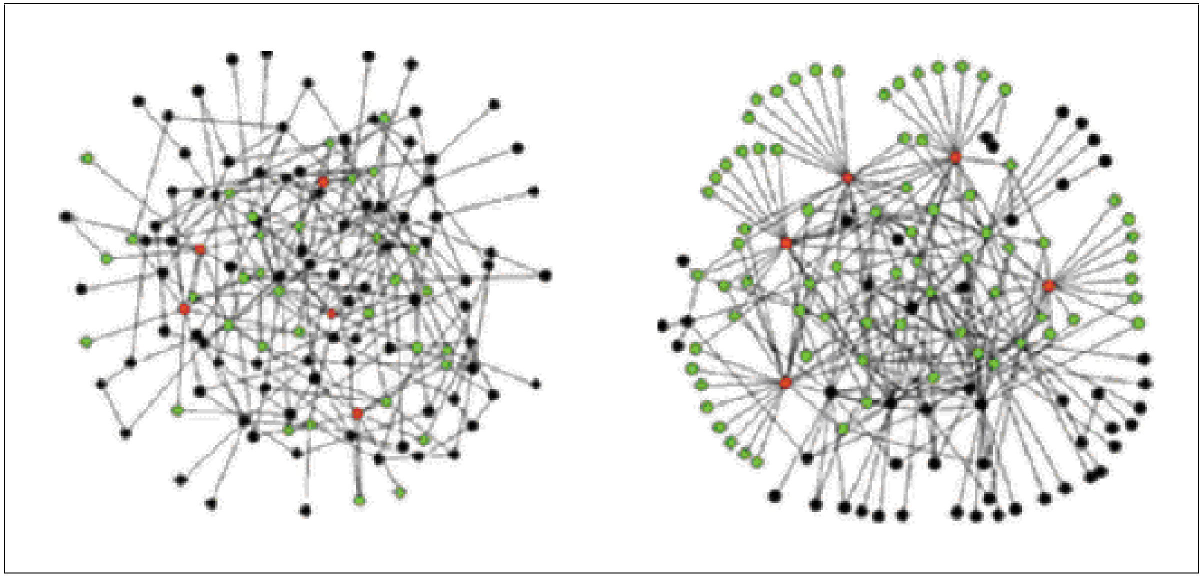


Figure A.4: Comparison of random and scale-free network topologies [Barabási 02b]

[14] The topologies in Figure A.4 were taken from Barabási [Cohen 02].

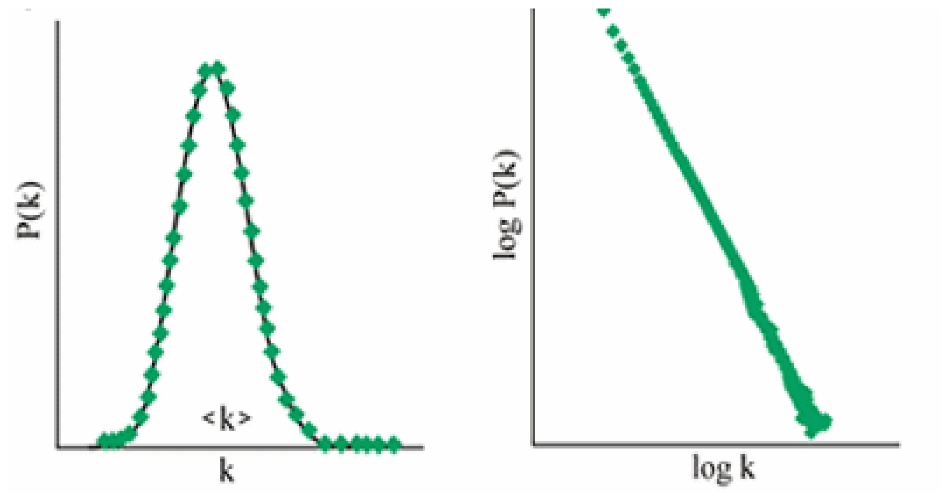


Figure A.5: Comparison of distributions for random and scale-free topologies [Barabási 02b]

Construction of Scale-Free Networks

The revelation that the Internet is scale-free has led to new attempts to understand the underlying mechanisms that make the Internet topology what it is. In particular the work of Barabási and colleagues [Barabási 02b, Willinger 02] has attempted to define some simple underlying mechanisms that result in the observed scale-free distribution of links. Barabási hypothesized that two simple rules were sufficient to define such scale-free networks:

- incremental growth (nodes are added one at a time and connected to the nodes that currently exist in the network)
- preferential attachment (new nodes are connected preferentially to existing network nodes that already have high numbers of links. There is also preferential attachment to nodes that are nearby.)

Networks thus constructed exhibit the required scale-free property.

While these results look encouraging, recent work [Chen 02, Willinger 02] has cast some doubt on whether these simple rules accurately capture the essence of the problem. This issue will be revisited later. There are now a number of software packages available that generate synthetic networks based on power law and other (e.g., random) statistics [Medina 01, SSFRN 02, ISI 02]. The software described in this report (GENeSIS) adds to this list.

Attacks Against Scale-Free Based Networks

Albert and colleagues have addressed the issue of how the Internet's topological properties affect its tolerance to failures and attacks. In the paper *The Internet's Achilles Heel: Error and Attack Tolerance of Complex Networks* [Albert 02a] they examine ability of networks to function under increasingly degraded conditions, specifically focusing on random failures and organized attacks. A significant conclusion they draw is that, for scale-free networks, random removal of nodes makes little difference to the ability of a scale-free network to route messages (the diameter changes little). This is consistent with observed robustness in the face of local failures of the actual Internet. However, with a targeted attack, the situation is significantly different. Targeted attacks would attempt to destroy the largest, most critical nodes (hubs), and in this situation, the effective diameter rapidly increases. With random networks the situation is different. Random networks do not exhibit the frequency of highly connected nodes that scale-free networks do. There is thus much less difference between the responses to random failures and organized attacks for networks with random topologies.

These differences are illustrated by the graphs shown in Figure A.6. These graphs were extracted from the paper mentioned above. Graph *a* shows the results of synthetic random (E) and scale-free (SF) networks under random failure and organized attack. The fraction of nodes removed is *f*, while *d* is the effective network diameter. These results show the following:

- the insensitivity of scale-free networks to random failure (squares)
- the greater sensitivity of scale-free networks to attack as opposed to random failure (circles and squares)
- the insensitivity of random networks to attack or random failure (diamonds and triangles)

Graphs *b* and *c* show similar characteristics for subsets of the actual Internet and the World Wide Web.

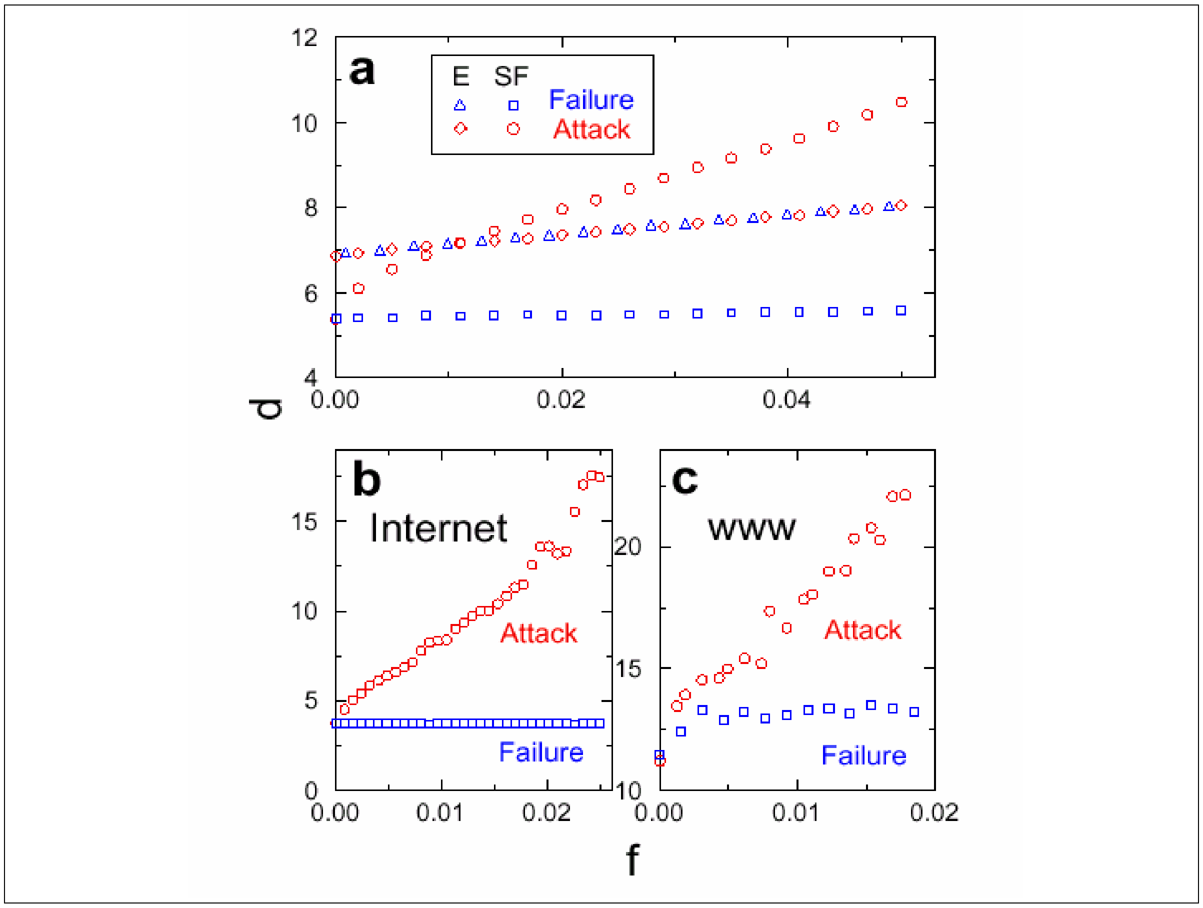


Figure A.6: Response of networks to random failures and organized attacks [Albert 02b]

Natural Versus Engineered Systems

It is empirically evident that many of the Internet's topological features scale using the power law [Faloutsos 99, Tangmunarunkit 01]. However, the reason it, and other systems, do so is still not universally agreed on. Carlson and Doyle claim that engineered systems (be they biological or man-made in origin) have designed-in features that make their behavior quite different from those of non-engineered systems studied by physicists [Carlson 02]. Such non-engineered systems exhibit properties that tend to be homogeneous and whose responses are ensemble averages over the system's individual particles. This is the case with the forest fire example. Carlson and Doyle describe engineered systems as having highly optimized tolerance (HOT) and show how their theory can explain these power law distributions. They claim that the examples physics commonly focuses on deal with self-organizing criticality around phase transition points and that engineered systems may operate far from phase transition.

As an illustration [Carlson 99] they extend the forest fire example to include an engineered component, firebreaks.[15] Without such barriers, a fire will spread through the forest (assuming the critical tree density has been exceeded). However, with judiciously placed barriers, the loss can be greatly reduced. Optimal placement of barriers can be determined based on cost-benefit analysis. Such a tradeoff can be determined through minimizing the expected cost J, where pi is the probability of an event i whose size is li, and where ri is the resource allocated to deal with the event [Carlson 00b]:

$$J = \Sigma \text{ pi li} \quad \text{given} \quad \text{li} = f(\text{ri}), \text{ and } \Sigma \text{ ri} \bullet R$$

Given certain additional assumptions, this minimization leads to a power law that relates the event probability to its size: pi ~ li-α. Firebreaks might be optimally engineered for a particular spatial probability of spatial spark distribution or tree density. If these parameters change, then the optimal performance is degraded. Thus HOT systems tend to be robust in the face of known events but fragile in the face of unpredicted events [Carlson 00a].

Survivability Implications

The power-law results described above are relevant, since they may indicate (per the arguments of Carlson and Doyle) that cost-benefit issues are important in determining the Internet's topology. This has definite survivability implications. We can ask whether there are other topological arrangements that are as efficient as the current one but more resilient to attack. Would such alternates be economically viable? How could we develop appropriate firebreaks?

One intriguing possibility of alternative design is that based on the Watts and Strogatz model. In this case there is local clustering with a few long-distance links. While the current Internet topology is vulnerable to the destruction of the small number of highly connected hubs, the Watts and Strogatz topology would be vulnerable to the destruction of the small number of long-distance links. Perhaps a topology that incorporates features of both models would be an improvement.

The Internet is robust against predicted events such as local router failure, but when unpredicted events occur, such as the release of a new virus, then fragility becomes evident. One event for which the Internet was not designed is a major loss of backbone routers. There will come a critical point at which communication across the network ceases. This type of massive failure is not one that the Internet has been designed to withstand. Thus HOT-designed features will not be applicable and we return to a system with a definite phase transition point. In such a case, percolation theory would become applicable again and we would need to look at survivability as a function of the density of disabled links.

[15] They also look at an avalanche example.

Appendix B: Running the GENeSIS Program

Data Input

Genesis runs in the Easel environment.[16] All data for a GENeSIS run are currently entered as an include file to the GENeSIS program. A data input file (for example, “data input.txt”) is specified through the include statement (include “::data files:data input.txt”) at the point in the GENeSIS program where the lines in the include file need to be located. (This point is identified near the foot of the GENeSIS program.) This external specification of data allows one to manage these data in appropriately named files. Only the included file name need be changed within the GENeSIS program before it is run. A typical set of data in an include file might look like this:

```
"run 1 - 500 node run", true, 500, 500, 4, 1.5, "",true, 0.25, 1.0, 1.0, 1.0, true, true,  # run 1
"run 2 - 1000 node run", true, 1000, 1000, 4, 1.5, "", true, 0.25, 1.0, 1.0, 1.0, true, true,  # run 2
“end”                                                                                   # terminator
```

Each data line corresponds to a separate node/network generation run. Thus individual generation runs can be stacked to run consecutively. Each line of the data contains 14 data elements although some of the data may be irrelevant for a particular run. For example, in the above run data, no network generation is called for in the first calculation (data element 7), so data elements 8 through 11 are not used. To terminate the run, the last line contains the title text string “end.” The data for each line contains the following:

1. A descriptive title for this run
2. GN (a Boolean value that determines whether to perform a nodal calculation or use node location data from a previous run)

   If GN=true then input values 3 through 6 are relevant:
3. number of nodes in node population
4. sampling size (see Section 3.6)
5. cell exponent c (determines the number of lowest level cells N =2^2c—see example in Section 3.3)
6. fractal clustering exponent (see Section 3.3)

   If GN=false then input values 7 and 8 are relevant:
7. file name of file that contains an existing nodal population
8. GL (a Boolean value that determines if a network is to be generated)

   If GL=true then input values 9 through 12 are relevant:

[16] Easel is available for download from http://www.cert.org/easel and runs on Macintosh OSX.

9. probability that if node A links with node B, and B with C, then node A will link with node C
10. preferential attachment (for link degree) exponent α (see Section 3.4)
11. preferential attachment (for distance) exponent σ (see Section 3.4)
12. probability that if node A [lb2]links to node B then node B links to node A
13. doSpatialPlots (if true then generate the nodal distribution and network plots)
14. doStatistPlots (if true then generate the statistical plots)

A complete record of a set of GENeSIS runs is automatically captured in the Easel output file as ASCII data. This includes a copy of the input data for each run and all output data described above. Note that any subsequent calculation will overwrite this file. If the data is important, its contents should be immediately transferred to another file. In order to use node data for network generation, that section of the output file that contains the node data (and only this data) should be copied and pasted into another ASCII file (for example, “node data.txt”). The name of this file (e.g., “node data.txt”) is then specified as the input file item 7 for the subsequent network generation run (GN=false).

Example Runs of GENeSIS

Figure B.1 shows example input to a GENeSIS run. The first 16 lines are comment lines and are simply there for clarification. As discussed earlier, the program automatically saves text output to the Easel text output file. The input data generate 25 nodes but do not link them.

```
#{ ######################################################################
            1: run title
            2: boolean: generate fractal node population
            3: number of nodes to be generated
            4: number of nodes to be sampled
            5: exponent for fractal box calculation
            6: exponent to vary degree of fractal clustering
            7: file name if node distribution to be read in
            8: boolean: network generation requested
            9: prob that if A links to B & B links to C, A will link to C
            10: exponent for node degree preferential attachment
            11: exponent for distance preferential attachment
            12: probability that if A connects to B, B connects to A
            13: boolean: display spatial plots
            14: boolean: display statistical plots
###################################################################### }#

"run 1 - 25 node run", true, 25, 25, 4, 1.5, "", false, 0.25, 1.0, 1.0, 1.0, true, true,
"end"
```

Figure B.1: Input to a small GENeSIS run

The resulting text output file is generated as shown in Figure B.2.

Plots of the spatial node distribution and fractal characteristics are also generated (Figure B.4 and Figure B.6). These figures are for illustrative purposes only—because of the small number of nodes used in this example, the distributions are statistically very poor.

If a network based on this distribution is subsequently desired, then the node data for nodes 0 through 24 inclusive is extracted from the output file and renamed (for example, as “node data.txt”). A second run can then be made using the generated node data. Of course, the two calculations can always be performed within the same run, but sometimes it is useful to separate them.

```
"run 2 - 25 node run", false, 25, 25, 4, 1.5, "node data.txt", true, 0.25, 1.0, 1.0, 1.0, true, true,
"end"
```

The resulting output is shown in Figure B.3, while the graphical plots are shown in Figure B.5, Figure B.7, and Figure B.8.

```
%%%%%%%%% Input data %%%%%%%%%
25 node run
node population generation requested
--- total number of nodes = 25
--- node sampling size = 25
--- cell box exponent = 6
--- clustering exponent = 1.5
spatial plots requested
statistical plots requested
%%%%%% End of input data %%%%%%%%%

generating node data..........................
node ID x-location y-location
------- ---------- ----------
0, 115.962, 153.337
1, 98.2674, 208.786
2, 437.554, 61.3323
3, 442.283, 82.7091
4, 469.218, 84.6452
5, 281.366, 26.484
6, 416.546, 72.4228
7, 255.526, 50.4265
8, 427.826, 43.5817
9, 459.94, 180.66
10, 466.395, 72.5001
11, 451.034, 61.7231
12, 18.2165, 49.2202
13, 164.666, 463.854
14, 454.196, 71.4427
15, 334.774, 88.7131
16, 459.646, 210.756
17, 435.884, 60.6086
18, 267.455, 474.849
19, 193.329, 486.155
20, 277.733, 487.1
21, 264.728, 9.89424
22, 62.0941, 59.5935
23, 76.8364, 63.6913
24, 102.968, 148.429
**********************************************************
computing fractal dimension data...
1/cell-length, # non-empty cells, log(1/cell-length), log(# non-empty cells)
---------- ------------------ -------------- ----------------------
1, 1, 0, 0
2, 4, 0.30103, 0.60206
4, 7, 0.60206, 0.845098
8, 15, 0.90309, 1.17609
16, 19, 1.20412, 1.27875
32, 23, 1.50515, 1.36173
------------------ Power law fit----------------------
exponent k for fit y = x^k is 1.06198
standard deviation on fit = 0.605369
**********************************************************
drawing node data...
```

Figure B.2: Output data from node generation run

```
%%%%%%%%% Input data %%%%%%%%
25 node run
prior node population read from file: node
data.txt
network generation requested
--- clique probabiluty = 0.25
--- weight on preferential attachment for link
degree = 1
--- weight on preferential attachment for
distance= 1
--- asymetric connection probability = 1
spatial plots requested
statistical plots requested
%%%%%% End of input data
%%%%%%%%%

reading node data.....
*************************************
*****
drawing node data...

*************************************
******
generating link data..........................

node ID x-location y-location IDs of linked
nodes
------- ---------- ---------- -------------------
0, 115.962, 153.337, 1, 2, 3, 4, 6, 10, 15, 24
1, 98.2674, 208.786, 0, 21, 24
2, 437.554, 61.3323, 0, 3, 11
3, 442.283, 82.7091, 2, 0, 4, 8, 11, 17, 18
4, 469.218, 84.6452, 3, 0, 5, 7, 9, 10, 14, 15,
16, 22
5, 281.366, 26.484, 4, 7, 9, 12, 22, 23
6, 416.546, 72.4228, 0
7, 255.526, 50.4265, 5, 4, 9, 12, 23
8, 427.826, 43.5817, 3, 11
9, 459.94, 180.66, 4, 5, 7
10, 466.395, 72.5001, 4, 0, 14, 16
11, 451.034, 61.7231, 3, 2, 8, 13, 18
12, 18.2165, 49.2202, 5, 7, 23
13, 164.666, 463.854, 11, 19
14, 454.196, 71.4427, 10, 4, 16, 22
15, 334.774, 88.7131, 4, 0, 22
16, 459.646, 210.756, 4, 10, 14
17, 435.884, 60.6086, 3
```

```
18, 267.455, 474.849, 3, 11, 20
19, 193.329, 486.155, 13
20, 277.733, 487.1, 18
21, 264.728, 9.89424, 1
22, 62.0941, 59.5935, 4, 5, 14, 15
23, 76.8364, 63.6913, 12, 5, 7
24, 102.968, 148.429, 0, 1
10000
*************************************
*
generating link degree data...
# nodes link degree log(# nodes) log(link
degree)
------- ----------- ------------ ----------------
5, 1, 0.69897, 0
3, 2, 0.477121, 0.30103
8, 3, 0.90309, 0.477121
3, 4, 0.477121, 0.60206
2, 5, 0.30103, 0.69897
1, 6, 0, 0.778151
1, 7, 0, 0.845098
1, 8, 0, 0.90309
1, 10, 0, 1
------------------ Power law fit------------------
exponent k for fit y = x^k is -0.918276
standard deviation on fit = 1.90487
************************************
lower dist upper dist cum. prob.
---------- ---------- ----------
0, 32.9117, 0.25
32.9117, 65.8235, 0.386364
65.8235, 98.7352, 0.409091
98.7352, 131.647, 0.431818
131.647, 164.559, 0.5
164.559, 197.47, 0.545455
197.47, 230.382, 0.636364
230.382, 263.294, 0.727273
263.294, 296.206, 0.772727
296.206, 329.117, 0.795455
329.117, 362.029, 0.886364
362.029, 394.941, 0.909091
394.941, 427.852, 0.931818
427.852, 460.764, 0.977273
460.764, 493.676, 1
************************************
drawing network...
*************************************
```

Figure B.3: Output from network generation run

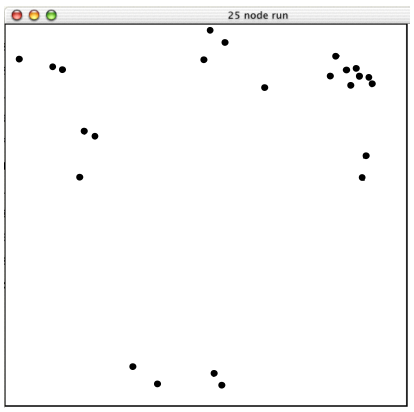


Figure B.4: A plot of the generated nodal distribution

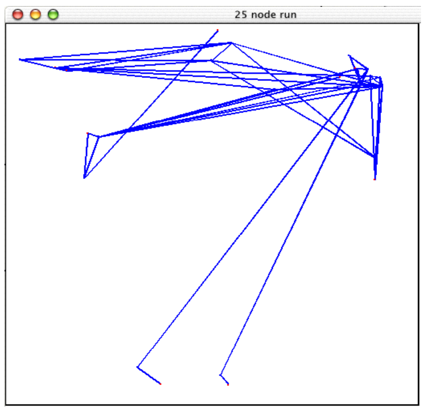


Figure B.5: A plot of the generated network

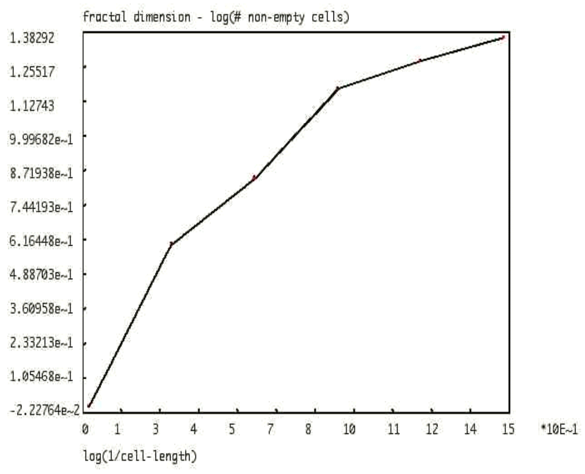


Figure B.6: Fractal dimension distribution for D_f

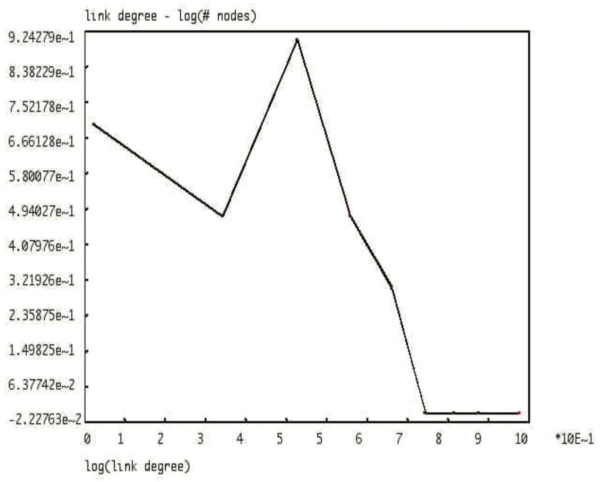


Figure B.7: Link degree distribution

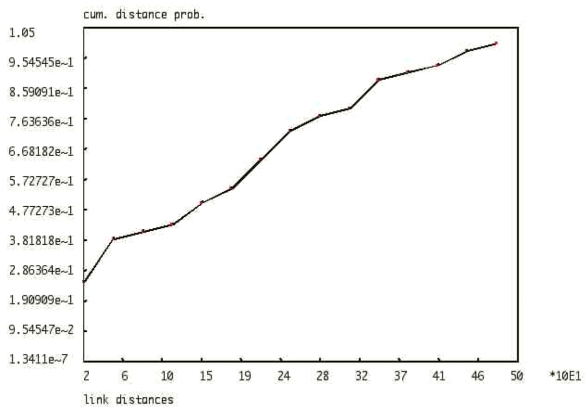


Figure B.8: Link distance distribution

References

As can be seen by many of the references listed below, this field has seen rapid growth in the past couple of years. Many articles have been published only this year. Most of these articles can be referenced through the Web, and appropriate URLs are given.

Albert 02a	Albert, R. & Barabási, A. “Statistical Mechanics of Complex Networks.” *Reviews of Modern Physics 74* (January 2002). <http://www.nd.edu/~networks/PDF/rmp.pdf>
Albert 02b	Albert, R.; Jeong, H.; & Barabási, A. *The Internet’s Achilles Heel: Error and Attack Tolerance of Complex Networks*. <http://citeseer.nj.nec.com/albert00internets.html> (2002).
Amaral 02	Amaral, Luis. *Complex Networks*. <http://amaral.chem-eng.northwestern.edu/Research/Old/Content_network.html> (2002).
Asundi 01	Asundi, J.; Kazman, R.; & Klein, M. *Using Economic Considerations to Choose Among Architectural Design Alternatives* (CMU/SEI-2001-TR-035, ADA399151). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu /publications/documents/01.reports/01tr035.html>
Ball 02	Ball, P. *Hubcaps Could Squash STDs*. <http://www.nature.com/nsu/010823/010823-6.html> (2002).
Barabási 99	Barabási, A. & Albert, R. “Emergence of Scaling in Random Networks.” *Science Magazine,* 286 (October 1999): 4509.
Barabási 02a	Barabási, A. *Diameter of the World-Wide Web*. <http://www.nd.edu/~networks/401130A0.pdf> (2002).
Barabási 02b	Barabási, A. *The Physics of the Web*. <http://physicsweb.org/article/world/14/7/9> (2002).

Baran 02	Baran, P. *On Distributed Communications: I. Introduction to Distributed Communications Network*. <http://www.rand.org/publications/RM/RM3420/ (2002).>
Callaway 00	Callaway, D.S.; Newman, M. E. J.; Strogatz, S. H.; & Watts, D. J. “Network Robustness and Fragility: Percolation on Random Graphs.” *Physical Review Letters 85,* 25 (2000): 5468-5471. <http://arxiv.org/abs/cond-mat/0007300/>.
Carlson 99	Carlson, J. M. & Boyle, J. “Highly Optimized Tolerance: A Mechanism for Power Laws in Designed Systems.” *Physical Review E 60,* 2 (1999): 1412-1427. <http://pre.aps.org/>.
Carlson 00a	Carlson, J. M. & Boyle, J. “Highly Optimized Tolerance: Robustness and Design in Complex Systems.” *Physical Review Letters 84,* 11 (2000): 2529-2532. <http://pre.aps.org/>.
Carlson 00b	Carlson, J. M. & Boyle, J. “Power Laws, Highly Optimized Tolerance, and Generalized Source Coding.” *Physical Review Letters 84,* 24 (2000): 5656. <http://pre.aps.org/>.
Carlson 02	Carlson, J. M. & Boyle, J. “Complexity and Robustness.” *Proc. Natl. Acad. Sci. 99,* 1 (February, 2002): 2538-2545. <http://www.pnas.org/cgi/content/abstract/99/suppl_1/2538>.
Chen 02	Chen, Q.; Hyunseok, C; Govindan, R.; Jamin, S.; Shenker, S.; & Willinger, W. “The Origin of Power Laws in Internet Topologies Revisited.” *Proceeedings of IEEE Infocom, 2002.* New York, N.Y., June 23-27, 2002. <http://citeseer.nj.nec.com/461619.html>.
Chen 93	Chen, S. et al. “On the Calculation of Fractal Features from Images.” *IEEE Transactions on Pattern Analysis and Machine Intelligence 15,* 10 (October 1993): 1087-1090.
Cohen 02	Cohen, D. “All the World’s a Net.” *New Scientist 174,* 2338, (April 2002): 24. <http://www.sciencenewsweek.com/firsts/NewScientist041302.htm>.
Dezsö 02	Dezsö, Z. & Barabási, A. *Halting Viruses in Scale-Free Networks* <http://xxx.lanl.gov/PS_cache/cond-mat/pdf/0107/0107420.pdf> (2002).

Doar 96	Doar, M. “A Better Model for Generating Test Networks.” *Proceedings IEEE Global Telecommunications Conference/GLOBECOM'96*, London, November 1996. <http://www.geocities.com/ResearchTriangle/3867/publications.html>.
Erdös 60	Erdös, P, Rényi, A. “On the Evolution of Random Graphs.” *Publication of the Mathematical Institute of the Hungarian Academy of Sciences 5* (1960): 17-61.
Faloutsos 99	Faloutsos, M.; Faloutsos, P.; & Faloutsos, C. “On Power-Law Relationships of the Internet Topology,” 251-262. *Proceedings of ACM SIGCOMM* Harvard, MA, September 1999. <http://citeseer.nj.nec.com/faloutsos99powerlaw.html>.
Frood 02	Frood, A. *The Weakest Link*. <http://online.sfsu.edu/~webhead/weakl.html> (2002).
Girvan 02	Girvan, M. & Newman, M.E.J. *Community Structure in Social and Biological Networks*. <http://www.santafe.edu/sfi/publications/wpabstract/200112077> (2002).
ISI 02	Information Sciences Institute (ISI), University of Southern California School of Engineering. *The Network Simulator*. <http://www.isi.edu/nsnam/ns/> (2002).
Jin 00	Jin, C.; Chen, Q; & Jamin, S. *Internet Topology Generator* (CSE-TR-443-00). Michigan: Dept. of EECS, University of Michigan, 2000.
Lloyd 01	Lloyd A., M. & May, R. M. “How Computer Viruses Spread Among Computers and People.” *Science Magazine*, 292 (2001): 1316-1317.
Magoni 02	Magoni, D. & Pansiot, J. *Analysis of the Autonomous System Network Topology*. <http://www.acm.org/sigcomm/ccr/archive/2001/jul01/ccr-200107-magoni.pdf> (2002).
Malamud 02	Malamud, B., D.; Morein, G.; & Turcotte, D.L. *Forest Fires: An Example of Self-Organized Critical Behavior*. <http://www.sciencemag.org/cgi/reprint/281/5384/1840.pdf> (2002).

May 01	May, R. M. & Lloyd, L. A. "Infection Dynamics on Scale-Free Networks." *Phys. Review E* 64 (2001): 066112.
Medina 00	Medina, A.; Matta, I; & Byers, J. "On the Origin of Power Laws in Internet Topologies." ACM *SIGCOMM Computer Communication Review 30,* 2 (April 2000): 18-28. <http://www.acm.org/sigcomm/ccr/archive/2000/april00/matta.pdf>.
Medina 01	Medina, A.; Lakhina, A.; Matta, I.; & Byers, J. "BRITE: An Approach to Universal Topology Generation." *Proceedings of IEEE International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems.* Cincinnati, Ohio, August 2001. <http://citeseer.nj.nec.com/452293.html>.
Montoya 02	Montoya, J. & Solé, R. *Small World Patterns in Food Webs*. <http://arxiv.org/PS_cache/cond-mat/pdf/0011/0011195.pdf> (2002).
Moreno 02	Moreno, Y.; Pastor-Satorras, R.; & Vespignani, A. "Epidemic Outbreaks in Complex Heterogeneous Networks." *The European Physical Journal B 26* (2002): 521-529. <http://www.ictp.trieste.it/~yamir/b01473.pdf>.
Newman 02	Newman, M.; Watts, D.; & Strogatz, S. *Random Graph Models in Social Networks*. <http://citeseer.nj.nec.com/445095.html> (2002).
Palmer 01	Palmer, C.; Siganos, G..; Faloutsos, M.; Faloutsos, C.; & Gibbons, P. "The Connectivity and Fault-Tolerance of the Internet Topology." *Proceedings of Workshop on Network-Related Data Management (NRDM 2001).* Santa Barbara, CA, May 2001. <http://citeseer.nj.nec.com/539145.html>.
Pastor-Satorras 01a	Pastor-Satorras, R.; Vázquez, A.; & Vespignani, A. "Dynamical and Correlation Properties of the Internet." *Physical Review Letters 87,* 25 (December 2001): 258701.
Pastor-Satorras 01b	Pastor-Satorras, R. & Vespignani, A. "Epidemic Dynamics and Endemic States in Complex Networks." *Phys. Review E 63* 6 (May 2001): 066117.

Pastor-Satorras 02a	Pastor-Satorras, R. & Vespignani, A. “Immunization of Complex Networks.” *Phys. Rev. E 65,* 3 (April 2002): 036104. <http://arxiv.org/PS_cache/cond-mat/pdf/0107/0107066.pdf >.
Pastor-Satorras 02b	Pastor-Satorras, R. & Vespignani, A. “Epidemic Spreading in Scale-Free Networks.” *Physical Review Letters, 86,* 14 (April 2002): 3200-3203. <http://citeseer.nj.nec.com/407844.html>.
Pastor-Satorras 02c	Pastor-Satorras, R. & Vespignani, A. “Epidemic Dynamics in Finite Size Scale-Free Networks.” *Physical Review E 65,* 3 (March 2002): 035108.
Patch 02	Patch, K. *Net Inherently Virus Prone*. <http://www.trnmag.com /Stories/032101/Net_inherently_virus_prone_032101.html> (2002).
SSFRN 02	SSF Research Network. *Scalable Simulation Framework*. <http://www.ssfnet.org/homePage.html> (2002).
Staniford 02	Staniford, S.; Paxson, V.; & Weaver, N. “How to Own the Internet in Your Spare Time.” *Proceedings of the 11th USENIX Security Symposium*. San Francisco, CA, August 5-9 2002. <http://www.icir.org/vern/papers/cdc-usenix-sec02/>.
Stauffer 94	Stauffer, D. & Aharony, A *Introduction to Percolation Theory, 2nd Edition*. London: Taylor and Francis, 1994.
Strogatz 01	Strogatz, S. “Exploring Complex Networks.” *Nature 410* (March 2001): 268-276 <http://tam.cornell.edu/SS_exploring_complex_networks.pdf>.
Stubna 02	Stubna, M. & Fowler, J. *An Application of the Highly Optimized Tolerance Model to Electrical Blackouts*. <http://www.chaos.cornell.edu/hot3.pdf> (2002).
Tangmunarunkit 01	Tangmunarunkit H., Govindan R., Jamin, S., Shenker, S. & Wollinger, W. *Network Topologies, Power Laws, and Hierarchy,* University of Michigan, USA, Technical Report USC-CS-01-746, 2001. <http://citeseer.nj.nec.com/465834.html>.

Tauro 01	Tauro, S.; Palmer, C.; Siganos, G.; & Faloutsos, M. "A Simple Conceptual Model for the Internet Topology." *Proceedings of Sixth Global Internet Symposium in conjunction with Globecom 2001*. San Antonio, Texas, November 25-29, 2001. <http://www.cs.ucr.edu/~michalis/PAPERS/jellyfish-GI.pdf>.
Turcotte 00	Turcotte, D.; Malamud, B.; Guzzetti, F.; & Reichenbach, P. "Self-Organization, the Cascade Model, and Natural Hazards." *Proc. Natl. Acad. Sci. 99*, 1 (February 2000): 2530-2537. <http://www.pnas.org/cgi/content/full/99/suppl_1/2530>.
Vazquez 02	Vazquez, A.; Pastor-Satorras, R.; & Vespignani, A. *Large-Scale Topological and Dynamical Properties of the Internet*. <http://arxiv.org/abs/cond-mat/0112400> (2002).
Watts 98	Watts, D. & Sorogatz, S. "Collective Dynamics of Small World Networks." *Nature 393* (1998): 440-442. <http://tam.cornell.edu/SS_nature_smallworld.pdf>.
Waxman 88	Waxman, B. "Routing of Multipoint Connections." *IEEE Journal on Selected Areas in Communications 6*, 9 (December 1988): 1617-1622.
White 98	White, S. "Open Problems in Computer Virus Research." *Proceedings Virus Bulletin Conference*. Munich, Germany, Oct 22, 1998. <http://www.research.ibm.com/antivirus/SciPapers/White/Problems/Problems.html>.
Willinger 02	Willinger, W.; Govindan, R.; Jamin, S.; Paxson, V.; & Shenker, S. "Scaling Phenomena in the Internet: Critically Examining Criticality." *Proc. Natl. Acad. Sci. 99*, 1 (February 2002): 2573-2580. <http://www.pnas.org/cgi/content/full/99/suppl_1/2573>.
Yook 02	Yook, S. *Modeling the Internet's Large-Scale Topology*. <http://arxiv.org/abs/cond-mat/0107417> (2002)

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE December 2002	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Network Survivability Analysis Using Easel		5. FUNDING NUMBERS F19628-00-C-0003
6. AUTHOR(S) Alan M. Christie		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2002-TR-039
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-2002-039
11. SUPPLEMENTARY NOTES		
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

The survivability of large, complex networks such as the Internet is an increasing concern, but such networks are difficult to analyze because they are topologically complex, highly non-linear in their responses, and inherently unbounded (i.e., no node in the network can have global knowledge). This report will describe how to develop statistically valid networks and, as an example of their use, apply them to the simulation of virus propagation. It will illustrate the construction of network topologies with GENeSIS, a program written in the general-purpose programming language Easel. The report will also summarize ongoing significant work in this area and give readers insight into how information or artifacts flow through networks and how networks respond to major disruptions.

14. SUBJECT TERMS network, security, network security, network topology, virus propagation, Easel, GENeSIS			15. NUMBER OF PAGES 70
16. PRICE CODE			
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL