

Peer-to-peer (P2P) Simulation for Network Security

Daniel O. Rice and George Wright
Loyola College in Maryland
4501 N. Charles Street
Baltimore, MD 21286 USA
{drice2, geo}@loyola.edu

Abstract—Peer-to-peer (P2P) networks have become a primary propagation mechanism of malicious code through file sharing applications. Many of the modern day malicious codes are being custom designed and deployed to specifically target P2P networks. These codes are tailored to take advantage of the characteristics of P2P protocols and the resulting P2P networks that typically overlay existing networks like the Internet. This research paper discusses an ongoing investigation into a method for increasing P2P networks’ resistance to malicious code propagation. The concept is shown through the simulation of P2P networks and the application of mechanisms that influence the topology of the network. We choose to demonstrate this on P2P networks because they have several important characteristics that may increase these networks’ vulnerability to malicious code attacks (these are some of the same characteristics that make P2P so valuable). P2P networks are self-organizing through users’ “sharing” choices, they are decentralized with a decreased emphasis on a central coordinating authority; and each node in a P2P network determines its own sharing parameters. This paper focuses on applying a simulation methodology to the creation of P2P networks using several Java classes. The model is validated by comparing simulation results to real-world P2P networks. P2P networks are simulated under two regimes. In the first, network nodes download files based on their experience only; the more traffic one node has with another, the more likely it is to seek downloads node. Under the second regime, a node chooses to do traffic with another depending on a price function. The price function in this work is linked to the Pearson coefficient, a measure that captures an important structural dimension of how the network is connected. The research in this paper concludes that the application of a pricing mechanism can lead to networks that evolve in a significantly different way. Ongoing research will determine if the application of a pricing, or other mechanisms, can create networks more resistant to the propagation of malicious code.

I. INTRODUCTION

Peer-to-peer (P2P) networks are a member of a much larger family of networks often referred to as distributed transient networks (DTNs). P2P networks exhibit behavior typically leading to three characteristics:

Self-organization through user choices in sharing of resources and services (files, storage, processor);

Decentralization due to the decreased emphasis on a central coordinating authority; and, Autonomy in that each node in a P2P network determines when and how much the node will make use of resources available on other nodes

and how available it will make resources that it possesses. [10]

It follows that P2P networks tend to be “more scalable, robust, and adaptive than other forms of distributive systems” and particularly “difficult to study due to their size and the complex interdependencies between users, application, protocol and network.” [12]

In the literature, P2P simulators have been developed to test networks and protocols. [12] However, many of these simulators have been designed to test very specific aspects of P2P networks. They do not capture even a fraction of the complexities inherent to these very large, diverse, and complex networks, nor are they amenable to validation. In this research we develop, test, and validate a network simulator that has been designed to capture the salient behavioral features of large complex P2P networks. Our immediate goal for this paper is to study the effect of pricing policies on the way network connectivity evolves. Our ultimate goal is to study the impact of security risks on these networks and the mitigating effects, if any, of pricing policies.

II. P2P NETWORK SECURITY – VIRUSES AND WORMS

In the first half of 2007, Symantec documented 212,101 new malicious code threats, a 185% increase over the previous period and a 318% increase over the first half of 2006. [11] These threats include trojans, worms, viruses, and backdoors. Symantec reports that of the malicious code that propagated in the first half of 2007, 22% did so through file sharing in general P2P networks, 18% in the Kazaa P2P network, 15% in the Morpheus P2P network, 15% in the eDonkey P2P network, and 5% in the Winny P2P network (shown in Fig. 1 below).

Rank	Propagation Mechanism	Percentage of Threats
1	File Transfer/Email Attachment	46%
2	File Transfer/CIFS	24%
3	File Sharing/Peer-to-Peer	22%
4	File Sharing/Executables	22%
5	File Sharing/Peer-to-Peer/Kazaa	18%
6	Remotely Exploitable Vulnerability	18%
7	File Sharing/Peer-to-Peer/Morpheus	15%
8	File Sharing/Peer-to-Peer/eDonkey	15%
9	File Sharing/Peer-to-Peer/Winny	5%
10	Backdoor/Kuang2	3%

Fig. 1. Propagation Mechanisms [11]

Also significant, as noted in the “Top ten new malicious code families” section of the Symantec report, contemporary “malicious code authors seem to be diversifying their propagation mechanisms by combining worms with a viral file-infection component.”

As pointed out in [11], the obvious recommendation to contend with this growing malicious code threat is that administrators should look to block this type of activity using more specific port blocking used by these applications at the network boundary and protocol filtering (e.g., block all Kazaa traffic). Additionally, where possible P2P applications are should not be permitted on corporate networks, and these enterprises should take measures to prevent P2P clients from being installed on any computers on the network.

However, while this may be possible (albeit difficult) in some corporate network environments, it does restrict the advantages of using P2P in these networks (namely connectivity and convenience). Moreover, some networks cannot restrict P2P file sharing because it is becoming an important and acceptable use for most users. In these networks administrators should encourage end users who download files from P2P networks to scan all such files with a regularly updated antivirus product in order to mitigate the very real threat of malware.

III. P2P MALICIOUS CODE PROPAGATION AND COMPLEX NETWORKS

The emergent behavior of complex networks impacts network topology. For example, Internet topology is at least partially determined by how nodes in the network choose to connect to each other.

One way to describe evolving network topology is by looking at the connectivity of the individual nodes in the network. For example, the topology of the Internet can be described as having a power law degree distribution. [4] This simply means that if you count all of the links going in and out of each node in the network (also called the “degree” of the node), the distribution of the degree for all the nodes in the network would be described by a power law.

Practically speaking, this means that most nodes in the network have a relatively low degree while a few nodes in the network will be of a relatively high degree. We can depict this graphically by plotting node degree (the count of in-out links for each node) versus frequency (the count of the number of nodes of a given degree). When the degree (x-axis) versus frequency (y-axis) plot is fitted with a curve, that curve is best described by a power law (if we are dealing with a power-law degree distribution network).

Another aspect of network topology is a concept called “assortative mixing.” [8] Assortative mixing refers to how nodes in the network preferentially attach to other nodes. If high degree nodes, nodes with a relatively high connectivity compared to other nodes in the network, have a preference

to attach to other high degree nodes, then the network shows “assortative mixing.” If high degree nodes tend to attach to low degree nodes the network shows “disassortative mixing.” The “assortativeness” of a network can be calculated, as the Pearson product moment coefficient r , for any network. [8] The Pearson coefficient, r , has been empirically calculated for many existing networks, including social networks and the Internet (for which $r \approx -0.189$). [4]

In this research, a power-law degree distribution and Pearson’s r were used to validate the properties of simulated P2P network topology and the model’s behavior.

IV. EPIDEMIOLOGY AND COMPLEX NETWORKS

The epidemiological literature is full of studies that address the spread of the human epidemic process; that is, the spread of biological diseases, in populations through the interactions of individuals. Many of these studies use mathematical models to capture the spread of disease in human social networks. [2][5][9] Networks provide a convenient model to describe the social systems and epidemic process by which biological diseases spread. Most of the epidemic studies assume a heterogeneous or random network model. However, malicious computer codes spread in environments better characterized by scale-free networks. The understanding of scale-free connectivity is essential to understanding how malicious code propagates in computer networks. [9]

Two of the simplest epidemiological models are the susceptible-infected-susceptible (SIS) model and the susceptible-infected-removed (SIR) model. The SIS model models individuals who are either infected or susceptible for infection while the SIR models three states including susceptible, infected, and removed indicating immunity or death leading to an individual’s removal from the network. The SIR model allows for individuals to be removed from the network in the case of death or acquired immunization. This fits our scenario, when infected computers are conceptually removed from the network when they either acquire immunity or die (e.g., from an unrecoverable crash). The SIR model will be applied in the simulation model in ongoing research into this area.

V. P2P SECURITY AND THE THEORY OF COMPLEX NETWORKS

The Internet and P2P networks have been shown exhibit approximately power-law degree distributions. These networks have also been shown to be extremely susceptible to virus and worm propagation. P2P viruses, such as the Kazaa, Grokster, iMesh and the Polip virus, are specifically designed to take advantage of P2P networks. [11]

In [6], the authors explore the “robust yet fragile” nature of the Internet. The Internet has evolved into a network that is generally unaffected by random component failures but is extremely vulnerable to targeted attacks.

Most social networks, on the other hand, are more robust and less fragile than the Internet. In [8], it is observed that "in social networks...diseases spread easily...however, this type of network has a small central set of people that the disease actually reach...they support epidemics...but the epidemic is limited in who it can reach."

It appears that social networks, and many other scale-free networks, are able to control epidemics organically through their structure. Interestingly, the network structure of the Internet and P2P render them much more vulnerable to the attacks of malicious code. Specifically, networks that have higher assortativeness, i.e., that have a higher Pearson's r , are more resistant to the propagation of infection. Networks with lower assortativeness, i.e., lower Pearson's r , are more prone to the propagation of infection. If a way could be found to influence the evolution of P2P networks' assortativeness toward higher Pearson's r values, the resistance of P2P networks to malware can be increased. Specifically, networks that have higher assortativeness, i.e., that have a higher Pearson's r , are more resistant to the propagation of infection. Networks with lower assortativeness, i.e., lower Pearson's r , are more prone to the propagation of infection. If a way could be found to influence the evolution of P2P networks' assortativeness toward higher Pearson's r values, the resistance of P2P networks to malware can be increased.

As previously mentioned the Pearson coefficient captures the level of assortative mixing in a network and is easily calculated for most networks. In this research, the Pearson's r is used in a pricing functions applied in the simulation to test how pricing in P2P can impact the topology of the network. In ongoing work simulation will be used to test a hypothesis that Pearson coefficient impacts the spread of viruses on the simulated P2P networks.

VI. SIMULATION

P2P network is similar to any other network simulation, except that it explicitly models the specific behavior of P2P networks mentioned above. That is, P2P network simulations model (1) self-organization of the network through users' "sharing" choices, (2) decentralization with a decreased emphasis on a central coordinating authority, (3) and each node in a P2P network determines its own sharing parameters. P2P network simulation must be validated to ensure they accurately model real world P2P networks. Specifically, the degree distributions of simulated P2P networks should be approximately power law. Additionally, the Pearson coefficient should be approximately -0.18 (approximating the Pearson of Internet, the network that P2P most commonly overlays).

The P2P simulator should also reflect the stochastic nature of real world P2P networks. This research presents a useful simulator that models P2P networks using several Java classes. The P2P network simulation model presented here is validated with respect to real-world P2P networks.

A. Approach

First, the P2P network simulation model is created using several Java classes, tested, and validated to ensure that it accurately reflects real-world P2P networks. Next, the simulation is tested under two regimes. In the first case, the network nodes download files based on experience, the count of files and the total number of bytes downloaded from a source. The more experience one node has with a source node, the more likely it is to seek more from that node. Under the second regime, a node chooses to download from a source node depending on a price function.

A Monte Carlo approach is taken where each model is simulated multiple times under both regimes and the results compared. The average Pearson's r is significantly different between the two regimes.

B. Model

The Java based P2P network simulator graphic user interface (GUI) is shown in Fig. 2 below.

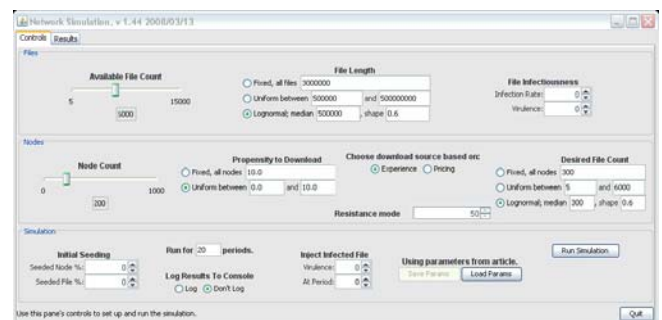


Fig. 2. Network Simulator Graphic User Interface

This figure provides a good basis for explaining the features and capabilities of the P2P network simulator. The first thing one might notice is that the GUI has two tabs, one for simulation controls, and the other for simulation results.

Simulation Controls Tab

This tab allows the user to set parameters and start the simulation. The simulator controls are grouped into three categories: Files, Nodes, and Simulation.

Files

This section of the GUI allows the user to set parameters for the simulated files. The Java class File represents the files that may be downloaded by nodes during a simulation run.

Table 1 shows the major attributes of the File class. Table 1 shows that each file has three major attributes.

TABLE 1
FILE CLASS DIAGRAM

File
infection : Infection
length : long
name : String

Infection

Each file has an Infection (another Java class, which may or may not be null). The major component of the Infection class is virulence. Virulence, which ranges from 0.0 to 100.0, is the percentage of uninfected files that an infected file will infect when it is introduced into a network node's directory. Here we mean "infection" to be an instance of malware that can spread across the network.

Length

This is the file's length or size in bytes. The GUI allows file length to be set in a number of ways. Our default is a lognormally distributed length.

Name

Each file has a unique alphabetic name, ranging from "aaaa" to "zzzz." The user can define the number of unique files available to the simulation run with the "Available File Count" slider. This control allows the user to provide enough distinct files to support a valid simulation while limiting the amount of memory required.

The "File Length" controls allow the user to specify one of three different ways for setting file length in bytes. This control is important because, under one simulation routine, network nodes choose to download file from another source node based on prior experience with the source node, where experience is quantified as files and bytes already downloaded from the source node.

The "File Infectiousness" controls determine the infection rate of the available files (from 0% to 100%) and the virulence of the infections that occur. When the stock of available files is generated at the beginning of the run, these controls affect how many files initially have a malware infection and how likely that infected file is to spread infection to other files in other nodes.

This and other class diagrams below are simplified. Only the attributes necessary for a quick explanation of the network simulator appear here.

Nodes

The middle segment of the control tab of the simulator GUI deals with nodes. A node is either a file-holding computer in a network or a user, depending on context. As a computer on the network, the node uploads or downloads files to or from other network nodes. As a user, the node makes decisions about what files to download and from

which other node the files are downloaded. Table 2 shows the six major attributes of the Node class.

TABLE 2
NODE CLASS DIAGRAM

Node
connectedness : int
countFilesDesired : int
directory : Hashtable<String, File>
experienceTable : Hashtable<String, Total>
identifier : String
propensityToDownload : double
resistance : double

Connectedness

Each node keeps track of its participation in a network in terms of the count of edges for which the node is either the beginning or the end. An "edge" is the line between nodes in a network, indicating that one of the nodes has had with the other. Connectedness is updated by a node method (not shown in Table 2) each time a file download occurs.

Count of files desired

On creation, each node is endowed with a desired file count. This simulates the fact that some nodes may want to download more files than other nodes. This defaults to a lognormal distribution.

Directory

Each node maintains a collection of its files. The collection is a Java hash table, keyed on the file's name.

Experience

Since one of the simulation regimes requires nodes to choose download sources based on prior experience, each node keeps knowledge of its experience downloading from other nodes. Experience is kept in a Total object (not discussed here) and keyed on node identifier.

Identifier

Each node has a unique alphabetic identifier, ranging from "Aaa" to "Zzz."

Propensity to download

This is the percentage of the file deficit (difference between files held and files desired) that a node will decide to download in a period. It ranges from 1.0 to 100.0.

Resistance

The probability that a node will notice that an incoming file is infected and disinfect it.

The node class is a threaded class, allowing each node to be simulated on a separately executing Java thread. The first node control is the node count slider. This allows the user to select a count of nodes that provides a valid simulation while conserving memory and limiting run time.

The control for propensity to download the user options for establishing each node's propensity to download files when the node is initially created. The radio buttons for method of choosing download source establish which of two regimes is simulated. Under the "Node experience only" regime, a node downloading a file chooses from among all possible sources of that file based on its historical experience. Experience, in the current implementation, is the product of count of files downloaded from a node and the byte-count downloaded from a node. A downloading node, under this regime, will always download from a node with which it has had the most prior experience. (If a node has had no prior experience with possible sources of a desired file, it will download from a source at random.)

Under the "Network pricing" regime, a node downloading a file chooses to download from the source ordering the desired file at the lowest price. Price, discussed at greater length elsewhere [needs reference], is determined by the equation:

$$\text{price} = 200 (20^{-\text{delta}R}) \quad (1)$$

where deltaR is the change, if any, in Pearson's r that would occur if the downloading and source nodes become connected. This pricing function influences the network to evolve with a greater Pearson's r value.

The "Resistance Mode" spinner control sets resistance of a node to infection, ranging from 0–100%. This is set stochastically from a triangular distribution with a modal value set by the spinner.

The controls for desired file count allow the user to choose from among three methods for establishing, when a node is created, how many files it will want to download. This simulates the fact that some network nodes want to download more files than others. It defaults to a lognormal distribution.

Simulation

The bottom segment of the control tab of the GUI deals with parameters of the simulation itself. The user can seed nodes with an initial complement of files, establish the count of periods to simulate, and choose to inject a single infected file of set virulence in a particular period. There are also controls to save and load a configuration of all user-settable parameters from a disk file, along with buttons to start the simulation and exit the program.

The seeding controls allow the user to start the simulation with a set percentage of nodes already holding a set percentage of the files each node desires. This simulates the fact that some nodes joining a peer-to-peer network may already have a directory of many files.

The run length control allows the user to set the count of periods to be simulated. The user can set the count to allow both the model's behavior to stabilize and the run time to be reasonable.

A logging control allows the user to observe the behavior of the simulation as it unfolds. The simulator code has many console print statements, some selectively commented out, that report each significant simulator event to the console device.

The next set of controls allows the user to inject a single malware-infected file, of set virulence, to be inserted into the directory of a randomly chosen node. The user can then observe the spread of infection, in terms of the percentage of nodes whose directories contain at least one infected file.

The buttons to load and save parameters allow the user to save the values of all settable parameters for later re-use. This has proved valuable as we sought a configuration that reliably mimicked the known behavior of peer-to-peer network evolution as observed in the real world.

The bottom line of the controls tab contains a status line and a button that quits the program entirely.

Results Tab

Fig. 3 shows the display of the simulation results from the configuration of parameters

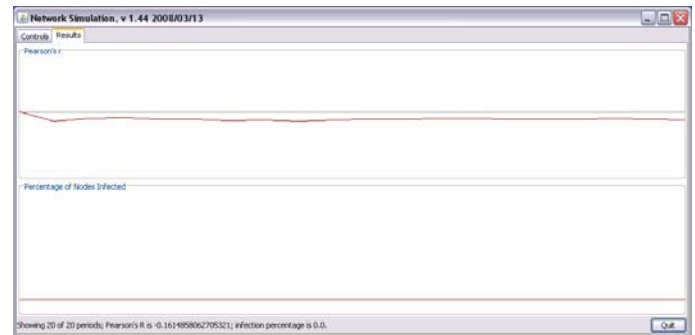


Fig. 3. Display of Network Simulation Results shown in Fig. 1.

Pearson's r

The top segment of the results tab shows how network interconnectedness, in terms of Pearson's r, has evolved during the simulation run. The x-axis varies to accommodate the count of simulation periods set by the user. The y-axis ranges from -1 to +1.

Percentage of Nodes Infected

The bottom segment of the results tab shows how infection evolves within the network. One way for malware

infection to enter the network is via files initialized with infection at the outset of the simulation. The other way is for a single infected file injected at random during the course of the run. The y-axis is the same as that of the Pearson's r display. The y-axis ranges from 0% to 100%, where the value is the percentage of nodes that have at least one infected file in their directory.

We do not explicitly explore the propagation of infection in this paper.

Status Line

The bottom line of the results tab is a status line and the usual quit button. The status line displays the current period's actual values for Pearson's r and infection percentage.

C. Simulation

When the user clicks the "Run Simulation" button on the controls tab of the simulator GUI, all the controls are disabled and the simulation begins. The following is sequence of events in the simulation.

Files are created. A master directory of all possible files, held by the simulator itself, is created and filled with files configured according to user-set parameters.

Nodes are created. A collection of nodes is created and filled with nodes configured according to user-set parameters.

A simulator instance, a Java threaded object, is created.

A network instance, another Java threaded object, is created, furnished with the collection of node instances, and installed as a member of the simulator object.

The simulator object is started. It immediately starts the network and waits for the network to complete its start-up.

The network, on start-up, starts each one of its nodes. The nodes are, at this point, not connected to each other at all yet.

After the network has started, control returns to the simulator object, which then simulates each period, up to the count of periods requested by the user. The activity that takes place in each simulated period is described below.

After all periods have been simulated, the final results (elapsed time of the run, final Pearson's r , final infection percentage, and matrix of node interconnectivity) are displayed. The model is set up for another run and the user controls are enabled.

The foregoing described the flow of control at the highest level of the simulation. We now must describe what takes place as the simulator thread, the master controller, simulates each period in turn. This is the sequence of events in each simulated period.

If the user has specified that an infected file be injected in this period, this specification is carried out. A node is selected at random. One of that node's files is selected at random. The selected file is infected as specified by the user.

The network then notifies each node to carry out its periodic behavior. The activity that each node carries out is described below.

When the network is notified that all nodes have completed their periodic behavior, the results display is updated for that period.

The real action of the simulator takes place as each node carries out its periodic behavior. Each node is a separately executing thread. When notified by the network that a new period has begun, each node's thread awakens and begins its periodic behavior, which consists of the following.

If the node's propensity to download is greater than a randomly generated number, it begins its downloading for the period.

If the node wants to download at all, it first checks to see how many desired files remain to be downloaded. If it still has files to download, it determines the percentage of the remaining files it wants this period (based on its propensity to download).

For each file it wants to download, the node chooses a file at random from the master directory of all available files.

If the node already has the desired file, it counts that file against its count of files to seek for the period and looks for the next file. If the node doesn't have the desired file, it asks the network for a list of all possible sources of the file. The network returns a list of all sources, sorted either by experience or by price, as specified by the user.

If there is a source of the file, i.e., if the list returned by the network has at least one entry, the node downloads the file from the top entry on the list, updating its directory, its connectedness, and its experience accordingly. Otherwise the node simply gets the desired file from the network's master list of available files, simulating the process of obtaining a file in some other way than from a P2P network.

When the node has downloaded all the files it's looking for this period (or if its propensity to download was less than or equal to the randomly generated number in step 1 above), the node decrements the count of still-active nodes. If this particular node is the last active node this period, the network is notified that all nodes have finished their behavior for the current period.

This completes the description of the simulator's behavior. At this point, the simulator GUI's control tab looks as it does in Fig. 1 and its results tab looks as it does in Fig. 2. Of course, the results will vary from run to run, because each run is stochastic. During each simulation run, each node is a separate thread of execution, pursuing its behavior according to the parameters set by the user. We explore the random nature of the runs in the next section.

VII. RESULTS

Validation of the simulation model requires that sufficient evidence that the behavior of simulation model depicts real P2P network behavior. In this section, we describe model validation and then discuss the comparison of P2P networks generated under the two schemes: experience and pricing.

D. Power Law Degree Validation

Fig. 4 below shows the results of 187 simulation runs of 1000 node networks for 40 periods. These numerical results from our network simulation show that our simulated networks exhibit scale-free characteristics and a power-law degree distribution. Specifically, the node degree distribution exhibits a power law (the R^2 coefficient is 0.6317).

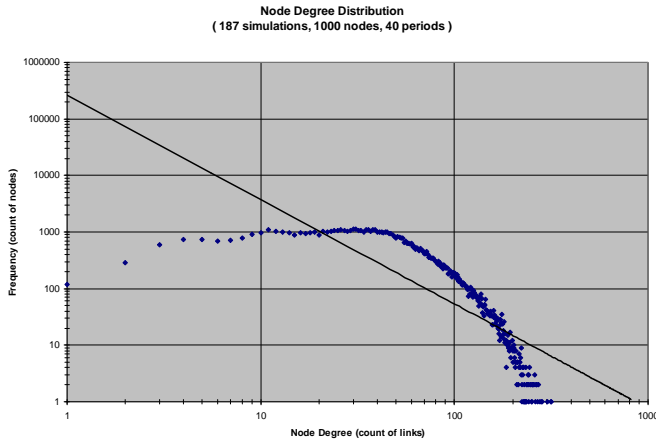


Fig. 4. Log-Log Plot Showing Power Law Degree Distribution

Simulation results show that the P2P simulation model degree distribution is approximately power-law, moving towards the validation of the model.

E. Assortativeness Validation

For validation purposes, we make the assumption that the Pearson of the P2P simulation model should approximate that of the Internet, where $r = -0.189$. [4] Numerous simulation runs have been used to determine that Pearson is highly sensitive to the configuration of user-set parameters in the simulation. The initial node seeding, the number of files that each node has at the beginning of the simulation, has the greatest impact Pearson's r , and experimentation showed that the hardest part of choosing an initial configuration was in setting node seeding. We would expect that some nodes entering a P2P network would already have a complement of desired files. Others would have few or none. Still others would have almost all the files one could expect them to want. But what combination of seeding (percentage of nodes seeded and percent of

desired file complement seeded) would yield a realistic Pearson's r ?

We used Monte Carlo simulation to answer this question. We using the general parameter configuration of Fig. 2, we simulated the model ten times each for various combinations of percent nodes seeded and percent files seeded. The percent of nodes seeded varied from 10% to 100% in increments of 10%; the percent of files seeded varied from 0% to 100% in increments of 10%. The results appear in Table 3.

TABLE 3
AVERAGE PEARSON'S R OVER 10 SIMULATIONS W/ VARIED SEEDING

		Percent Files Seeded										
		0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Percent Nodes Seeded	10%	-0.141	-0.144	-0.179	-0.228	-0.265	-0.306	-0.312	-0.387	-0.391	-0.451	-0.477
	20%	-0.142	-0.123	-0.131	-0.164	-0.211	-0.247	-0.272	-0.301	-0.349	-0.365	-0.406
	30%	-0.125	-0.117	-0.115	-0.127	-0.134	-0.182	-0.186	-0.217	-0.236	-0.246	-0.274
	40%	-0.135	-0.128	-0.108	-0.117	-0.127	-0.126	-0.143	-0.163	-0.179	-0.194	-0.206
	50%	-0.139	-0.145	-0.119	-0.117	-0.110	-0.122	-0.143	-0.151	-0.148	-0.198	-0.209
	60%	-0.139	-0.164	-0.141	-0.107	-0.114	-0.118	-0.144	-0.163	-0.166	-0.180	-0.183
	70%	-0.139	-0.165	-0.135	-0.133	-0.133	-0.124	-0.139	-0.160	-0.206	-0.202	-0.201
	80%	-0.148	-0.173	-0.150	-0.123	-0.126	-0.115	-0.154	-0.166	-0.163	-0.188	-0.217
	90%	-0.146	-0.188	-0.167	-0.132	-0.128	-0.126	-0.154	-0.149	-0.197	-0.171	-0.206
	100%	-0.134	-0.194	-0.173	-0.132	-0.120	-0.129	-0.163	-0.167	-0.182	-0.226	-0.253

Our objective is to find a seeding configuration that produces valid results, to wit, a Pearson's r of about -0.189. It appears from the table that this is about the average result we get with 30% of the nodes seeded with 60% of their desired file complement.

Another way of looking at the data in Table 3 is the surface shown in Fig. 5.

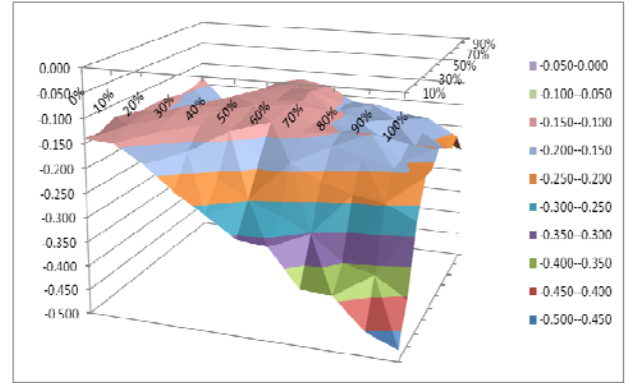


Fig. 5. Average Pearson's r Surface over Ten Simulations with Varied Seeding

Fig. 5 shows the contour of valid initial seeding configurations as light blue (-0.200—-0.150).

F. Application

The configuration shown in Fig. 2 is used with the seeding at the parameter level shown in Table 3 (shown highlighted in yellow, bolded and underlined in Table 3, where 30% of the nodes seeded with 60% of their desired files) to compare the evolution of a P2P network under

experience-based selection and price-based selection. We simulated the configuration 30 times, resulting in a mean Pearson's r of -0.190, standard deviation 0.0326.

The hypothesis that the mean Pearson's r is the nominal -0.189, cannot be rejected. However, is not alarming because exactly matching the Pearson is not absolutely necessary, the goal is to get close which is why that configuration was chosen. The z -value is $z = -0.175$.

However, more importantly are the results when the configuration as in the paragraph above is used in the pricing case, when the "Pricing" radio button in the "Nodes" section of Fig. 2 checked. That is, the nodes are now choosing to download files from peer nodes based not on experience but on choosing a source node with the lowest value of the price function. As before, we simulated the configuration 30 times, resulting in a mean Pearson's r of 0.107, standard deviation 0.0265.

The hypothesis that the mean Pearson's r under experience-based choice, -0.190, is the same as the mean Pearson's r under price-based choice, 0.107 is rejected ($z = -38.737$). We conclude that a pricing function that charges more for download choices that decrease the Pearson's r of a P2P network influences node behavior as desired.

G. Continuing Research

The model and set of initial configuration parameters have shown under simulation to faithfully reproduce the behavior of P2P networks on the Internet, when node choice of download source is made on the basis of prior experience. Furthermore, there is confidence exhibited in the model that a pricing function devised to increase the Pearson's r of the evolving network will shape the network as expected. Prior research predicts that that propagation of malware will be less on a price-based network than on an experience-based network. This is what we intend to investigate next.

REFERENCES

- [1] R. Anderson, R. May, and B. Anderson, *Infectious Diseases of Humans: Dynamics and Control*, Oxford University Press, Oxford, UK, 1992.
- [2] N. Bailey, *The Mathematical Theory of Infectious Diseases*, Hafner Press/MacMillan Pub. Co.; 2nd edition, London, 1975.
- [3] D. Chang, and C. Young, *Infection Dynamics on the Internet*, Computers and Security, Vol. 24, p. 280-286, 2005.
- [4] Q. Chen, H. Chang, R. Govidan, S. Jamin, S. Shenker, and W. Willinger, *The origin of power laws in Internet topologies revisited*, INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, New York, 2002.
- [5] O. Diekmann and J. Heesterbeek, *Mathematical Epidemiology of Infectious Diseases: Model Building, Analysis and Interpretation*, John Wiley & Sons, New York, 2000.
- [6] J. Doyle, D. Alderson, L. Li, S. Low, M. Roughan, S. Shalunov, R. Tanaka, and W. Willinger, *The "Robust Yet Fragile" Nature of the Internet*, California Institute of Technology Working Paper, obtained March 2005.
- [7] F-Secure Virus Descriptions : P2P-Worm, F-Secure Computer Virus Information Pages, <http://www.f-secure.com/v-descs/p2pworm.shtml> accessed on November 17th, 2005.
- [8] M. Newman, Assortative Mixing in Networks, *Physical Review Letters*, Vol. 89, No. 20, November 2002.
- [9] R. Pastor-Satorras, and A. Vespignani, *Epidemics and Immunization in Scale-Free Networks*, published in the Handbook of Graphs and Networks: from the Genome to the Internet, eds. Bornholdt, S., and Schuster, H., Wiley-VCH, Darmstadt, Germany, 2003.
- [10] D. Schoder, K. Fischbach, and C. Schmitt, *Core Concepts in Peer-to-Peer (P2P) Networking*, in R. Subramanian and B. Goodman (eds.): *P2P Computing: The Evolution of a Disruptive Technology*, Idea Group Inc, Hershey, 2005.
- [11] Symantec Internet Security Report: Trends for January 06 -- June 06, Volume X, Published September 2006, editors D. Turner and S. Entwisle, available at www.symantec.com.
- [12] N. Ting and R. Deters, 3LS - A Peer-to-Peer Network Simulator, Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P'03)