# Practical Applications of Computer Viruses

by Dr. Frederick B. Cohen

Computer viruses can create many problems for computer systems with inadequate integrity controls, and have caused widespread damage all around the world, but they can also provide a powerful parallel processing mechanism which could have dramatic impacts on how we use computers. Several practical computer viruses are already in everyday use, and new applications are being developed all around the world. Computer viruses may form the basis of a new model of computation with computer networks acting like a digital primordial soup, and viruses acting as genetically engineered living creatures which achieve goals by their combined actions.

‡ This research was funded by ASP, PO Box 81270, Pittsburgh, PA 15217, USA

## Reliable, Efficient Parallel Processing

On March 22, 1991, the world high speed computing record was broken by a Massachusetts company specializing in parallel processing. The previous record holder, contrary to popular belief, was Robert Morris's Internet Virus, designed to break into thousands of computers in the Internet *(Special issue of the Communications of the ACM, V?#?, 1989)*. This virus performed about 32 million operations per second on each of 6,000 computers, and another 3.2 million operations per second on each of 60,000 computers, for a grand total of 384 Billion operations per second! For pure processing cycles, computer viruses are some of the fastest distributed programs we know of, but unfortunately, many of them are malicious!

The same issues that make viruses a serious threat to computer integrity *(F. Cohen, "A Short Course on Computer Viruses", ASP Press, PO Box 81270, Pittsburgh, PA 15217, USA, 1990)* make them a powerfull mechanism for reliable and efficient distribution of computing. They distribute freely, easily, and evenly throughout a computing environment; they provide for general purpose computerized problem solving; and they are very reliable even in environments where computer systems fail quite often.

The use of self-replicating programs for parallel processing is not new. In fact, John von Neuman, one of the pioneers of the computer age, described reliable self-replicating programs in the 1940s. *(J. VonNeumann, "The Complete Works of John von Neumann")* In many early works, the 'living' computer program was not just a distant possibility, but the intent of the exercize. Early papers talked of 'making reliable organisms out of unreliable organs', and 'intelligent self-organizing systems with numerous components'.

## The Worm Programs

In 1982, Shoch and Hupp reported a series of successful experiments with parallel processing using self-replicating programs that spread through the Xerox computer network. *(J Shoch and J Hupp, "The 'Worm' Programs - Early Experience with a Distributed Computation", CACM pp172-180, March, 1982.)*

These so-called 'worm' programs would install themselves on computers which were not in use, performing 'segments' of the parallel processing problem being solved. Whenever an employee wanted to use their computer, they simply pressed the reboot button, and normal operations would resume. During the day, the worm would be trimmed back, running only on a few computers that were not in use, but at night, it would become active all over the United States, performing tens of millions of useful calculations per second.

Unfortunately, a bit-level error in one copy of the worm ended their experiments by causing the global Xerox network to reboot to the worm instead of the normal operating system. The entire network had to be restarted, and further experiments were barred.

## Time's a Wastin

It turns out that in most of the world's computers, there is more unused processing time than used processing time. If a more reliable mechanism had been used in these early experiments, we might now have thousands of very reliable high speed parallel processing applications using this vast amount of unused computer time for usefull purposes. One such application is the 'Lenstra-Manasse creature' which uses networked computers to factor large numbers. (A. Lenstra and M. Manasse, "Factoring by Electronic Mail" Eurocrypt, '89, appearing in "Advances in Cryptology", pp355-371, Springer Verlag, 1989.)

This virus uses an automated computer mail system to get permission from computer owners to cooperate in distributed processing, thus avoiding the problem of undesirable replication. Each 'segment' of this virus reports partial results back to a central location via electronic mail, and the central location then distributes new problems, again using electronic mail for communication. Although this virus has not achieved extremes in parallel processing performance, it is a good example of a successful virus that uses otherwise wasted resources for something useful. It displays the virus properties of replication, evolution, self-distribution, reliability, and high performance problem solving, while avoiding many of the problems of malicious viruses.

## Creatures in an Environment

This example also demonstrates a very simple example of the interaction between the environment and the virus in viral computation. This virus could not succeed if the environment wasn't made up of many cooperative researchers who give permission for it to operate. Just as biological systems often rely on diverse species cooperating with each other for mutual survival, computing environments with viruses rely on cooperative techniques in order to assure mutual survival.

## The Viral Bill Collector

Another useful computer virus is the automated 'bill collector', a virus that operates in a specially designed computing environment. The environment 'births' a new 'bill collector' every time a bill is to be collected (based on human interaction), kills 'bill collectors' whenever a bill is finally paid (people enter data on payments), and allows the bill collector to write letters and evolve through its life-cycle. In its simplest form, the bill collector is just a simple program that collects a single bill by sending a series of letters over time which may vary depending on whether the debtor is making payments or not.

## Simple to Write

It turns out that writing a computer program to collect a single bill is not very hard to do. In fact, in about an hour, a programmer can write a simple bill collecting program that will do a pretty good job of collecting a single bill. Instead of writing a complex database management system with an equally complex set of specialized routines for bill collection, we write a very small and simple bill collector, which replicates to achieve a complex global behavior.

## System Evolution

As we develop better and better bill collectors, we can start birthing them for collecting new bills, but we don't have to worry about all of the old bill collectors; eventually they will die out when their task is done. Thus the system can be made operational very quickly, and over time, evolve into a better and better system. With a little more effort, we can design bill collectors that evolve over time so as to improve their efficiency at bill collection.

## Random Variation and Selective Survival

In the simplest form, we can use pseudo-random variables to change the weighting of different collection strategies, and only replicate bill collectors that exceed a certain threshold of success. If we use less variation on bill collectors that are more successful, we will tend toward local optima. To attain global optima, we must occasionally use enough randomness to shake loose from the local optima. Over time, we may find several local optima, each with a fairly stable local population of bill collectors. Thus different species of bill collectors may coexist in the environemnt if there are adequate local niches for their survival. Cross breeding of species is feasible by taking select parameters from different species to birth new bill collectors. Given a proper distribution of randomness in the process, the system as a whole will tend toward more and more optimal bill collectors. This is commonly called "random variation and selective survival", and is roughly the equivalent of biological evolution as we now commonly speak of it.

## Birth and Death Processes

The "birth/death" process is central to the problem of designing viruses that don't run amok, as well as to the evolution of viral systems over time. If it weren't for the death of old bill collectors, the system would eternally be collecting bills under the original design. A global modification of all of the existing bill colectors would be required to make a system change, and this might be very hard to accomplish in a complex network. Birth and death are vital to system optimization in viral systems where the environment changes with time, since what is optimal today may not even survive tomorrow. This may even explain why biological organisms that live in relatively dynamic environments (e.g. people on land) have limited lifespans, while organisms that live in static environments (e.g. coral in an ocean) can live indefinately without noticeable ageing. Imagine how different human society would be if we still had Niandrethal people around.

## Complex Systems from Simple Interacting Processes

The behaviors we are discussing are getting complex, involving local and global optimization, evolution over time, and even the existence and cross-breeding of competitive species in an environment; but the computer programs we are discussing are still quite simple relative to other automated bill collectors. A typical automated bill collection system in use today is made up of from 500 to 10,000 pages of computer program, takes from 0.5 to 10 programmer-years to write, is unable to automatically move toward any sort of optimization of the collection process, and requires massive database modification to change the collection process. The viral bill collector takes only a few pages of program code, was written and operating in under 1 day, is able to attain local optima over time, and can be evolved during normal operation. The reason for this dramatic difference is that complex behavior can result from the interaction of simple mechanisms.

## Generating Sets

In the case of the viral bill collector, we create a small "generating set" of instructions which creates a complex system over time through birth/death processes, selective survival, and the interaction of coexisting species in the environment. Even quite simple generating sets can result in very complex systems. In fact, most of the interesting properties of systems resulting from generating sets such as these are undecidable. That means that only well designed generating sets operating in well understood environments will act with a great deal of predictability.

## Communications Mechanisms

One of the ways we can design predictable viral systems is by adding communications. Completely deaf and dumb viruses have a hard time surviving because they tend to be born and die without any controlling influences, and we get unstable situations which either consume too many resources and ruin the ecology or die from lack of sufficient biomass. With even rudimentary communications, viruses can survive far better. For example, real-world computer viruses survive far better if they only infect programs that are not yet infected. Too much communication also makes viruses inefficient, because they have to address an increasingly global amount of information. We suspect that communications is beneficial to viral survival only to the extent that it helps to form stable population relative to the resources in the environment.

## Artificial Life

The systems we have described may someday be widely recognized as rudamentary forms of artificial life, but for the moment, there is no general concensus among the scientific community about what constitutes life, and thus no widely accepted way to determine what constitutes artificial or natural life. There is however a formal definition of computer viruses which has been proposed as a formal definition for information based life. The formal definition addresses the issue of replication and evolution in an environment, and is quite inclusive.

## The Computer Virus Contest

The possibilities for practical viruses are unbounded, but the possibilities are only starting to be explored. Unfortunately, viruses have gotten a bad name, partly because there are so many malicious and unauthorized viruses operating in the world. As of early 1991, there were over 500 real-world viruses, with a newly designed virus being detected more than once per day worldwide. In an attempt to stem the tide of malicious and indiscriminant viruses, ASP has offered a cash award to the most useful virus written in each year.

The concept is simple; instead of writing malicious viruses, damaging other people's computer systems, hiding your identity, and risking arrest, prosecution, and punishment; virus writers have a legitimate venue for expressing their intellectual interest, and can get both recognition and financial rewards for their efforts. Since truly useful viruses may be competitive in the open market, contestants are offered the oportunity to license their creations to interested parties for still further financial reward.

The contest rules dissallow viruses that have been spread into the general environment, viruses placed in systems without explicit permission of the owner, and viruses without good mechanisms to control their spread.

## Summary and Conclusions

## The Author

Dr. Frederick B. Cohen is an independent researcher and educator living in Pittsburgh, PA. He is best known for his pioneering work in computer viruses, and the development of computer integrity mechanisms now in widespread use.