STAT 753
Computer Intrusion Detection

Computer Viruses, Epidemiology, and Worms

Lecture 8

Material is Courtesy of David Marchette

Course Outline

- Basics of TCP/IP Networking
- Network Attacks
- Pattern Recognition and Data Analysis
- Network Statistics
- Host Attacks
- User Profiling
- Computer Viruses and Worms
- Trojans and Covert Communication

Outline

- Computer Viruses
- Virus Epidemiology
- Immunology Revisited
- Computer Worms
- Some Famous Worms

< > - +

Viruses vs Worms

- I call a malicious program a virus if it infects other programs.
- I call it a worm if it spawns copies of itself (causes copies of itself to execute).
- Other definitions are possible. Some make the distinction that a virus requires user assistance to propagate (you need to run the infected program) while worms do not.
- Note that a program can be both a virus and a worm.

< > - +

How Viruses Work

- A virus changes the code in a program so that when the program is run the virus program is run instead (or in addition to the program’s code, if it’s clever).
- Boot viruses copy themselves onto the boot partition of a disk, so that when the computer is booted, the virus executes.

How Viruses Work

- A virus may simply make copies of itself onto programs, or it may do other nasty things:
  - Erase files.
  - Mail your passwords to its author.
  - Mail itself to your Friends.
  - Open up a backdoor to your computer.
- The possibilities are endless.

Virus Hoaxes

- There have been a number of virus hoaxes perpetrated over the years.
- A famous one was that a certain email had a virus and if you read the email you would get infected.
- This was hilarious, since everyone knew that you couldn’t get a virus by reading email.
- Those of you who are chuckling at our naiveté are wrong, you cannot get a virus by reading email.

< > - +

Virus Hoaxes

- Unfortunately, most mail readers will happily execute parts of your email for you, since this makes email so much more fun!
- This is why email viruses are no longer hoaxes

< > - +

More Virus Hoaxes

- There are still hoaxes going around. These cost people time, without requiring any work on the part of the hoaxer.
- My favorite, although really a joke rather than a hoax, was the “low tech” virus, where the email informed you that you had received a virus, but due to the low level of technology in the author’s country, the virus required help from you. So, please delete all the files from your disk.

More Virus Hoaxes

- One hoax actually claimed that you (the user) could contract a virus from reading an email. This functionality won't be available from your favorite mailreader software for at least a year, but you can bet they are working on it.
- The bottom line: don't execute things unless you know they are clean. This means in particular, set your email reader to the lowest functionality it will support. Ascii is your friend.

More Virus Hoaxes

- Note: if you receive email informing you that you have been infected and to delete the following files, DON'T DO IT!

Some Virus Facts

- There are thousands of computer viruses, more every day.
- The vast majority of them infect Microsoft Windows.
- They are becoming more sophisticated every month.
- They cause untold billions of dollars in lost time. Actually, they will tell you how much, but they make it up. Quantifying “lost productivity” is very difficult.

Why Pick on Windows?

- MS is the dominant OS, and hence the most effective target.
- MS is committed to making an OS that is easy to use and user friendly. This ease of use and friendliness makes it a great environment for viri.
- This focus on ease is at the expense of a focus on security. (A very secure system is not much fun to use.)

< > - +

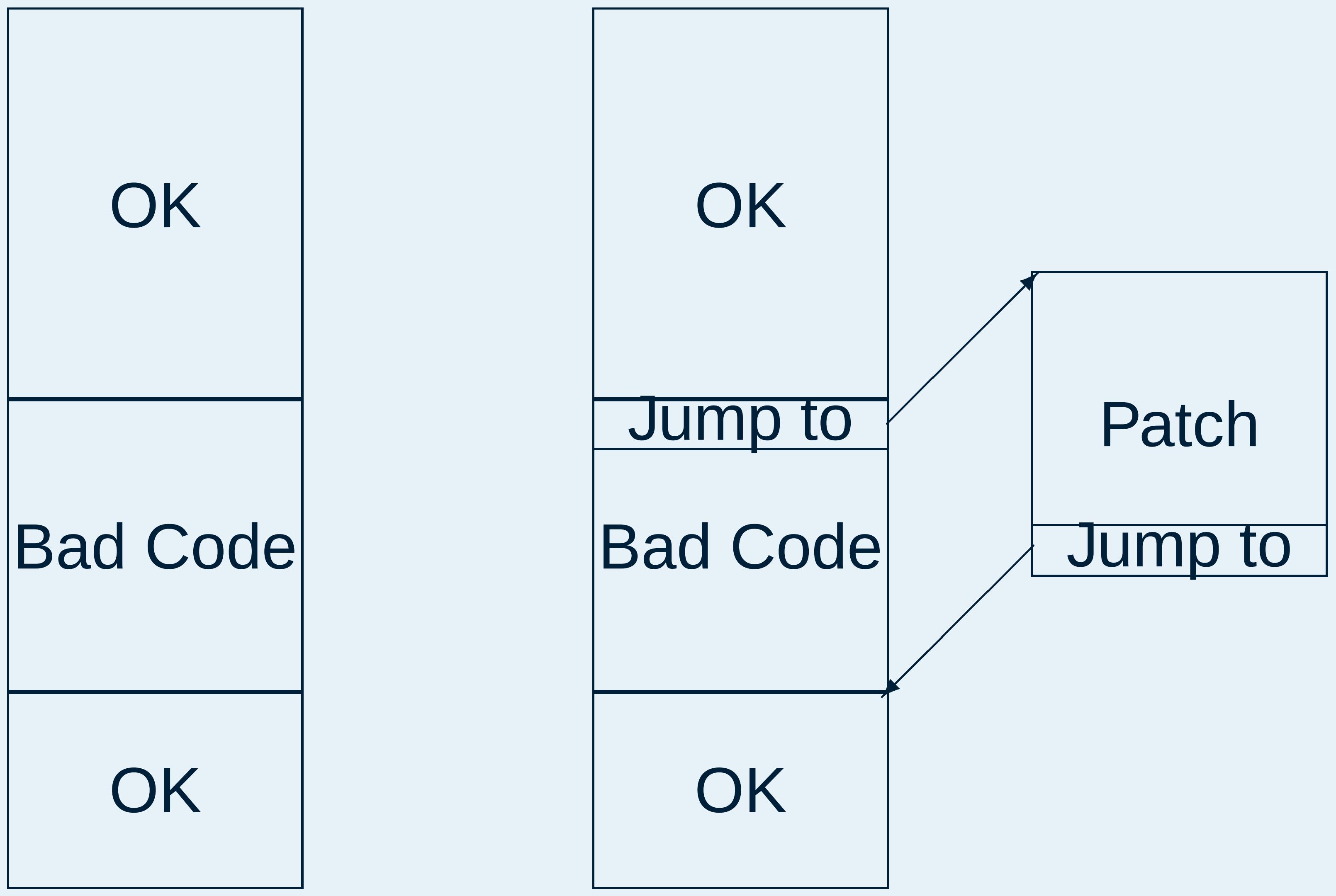
Why Pick on Windows?

- Only recently has MS become serious about security.
- I bought a Windows box last year.
- The default permissions for users is administrator, no password.
- This is not a good sign.

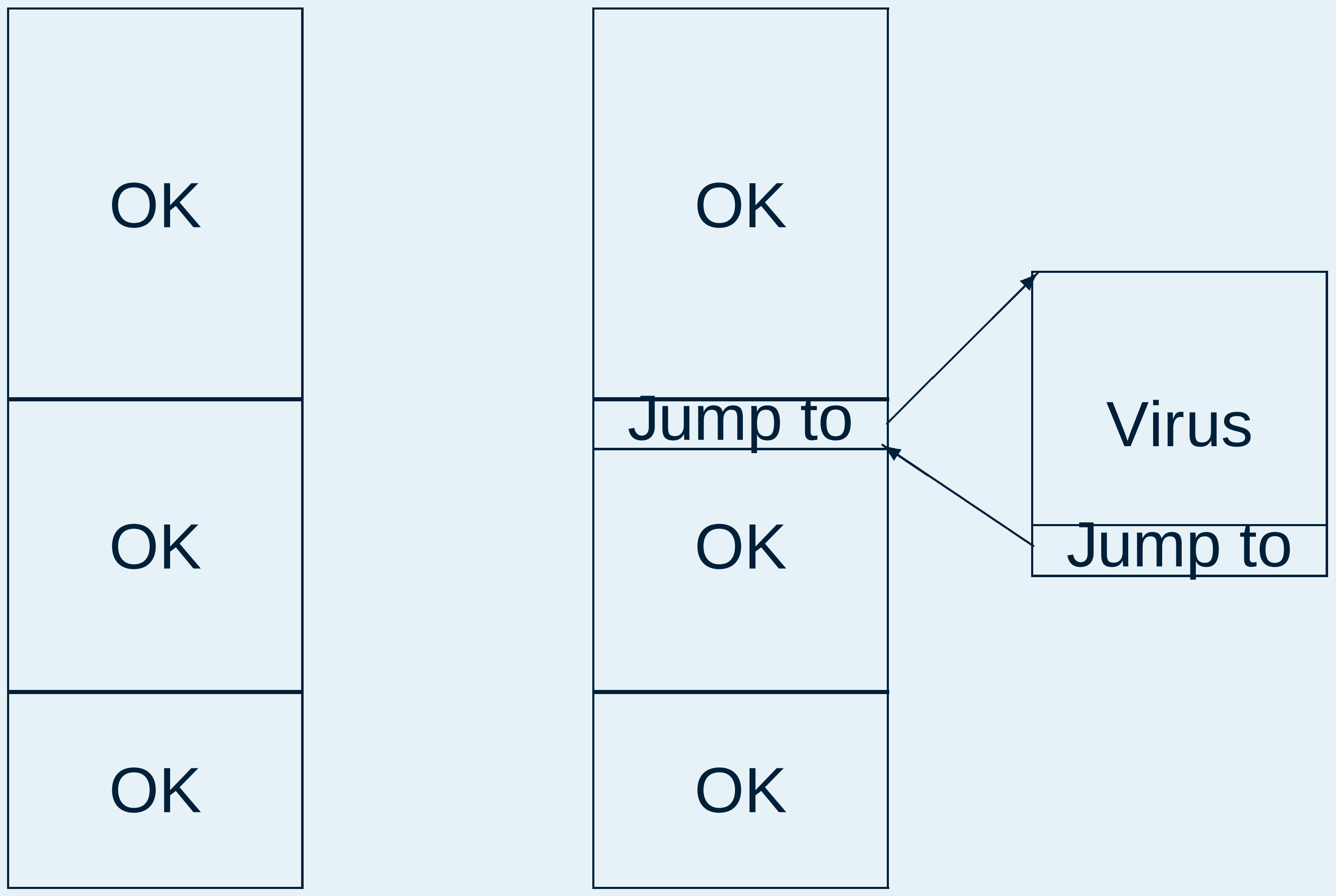
Virus Replication

- Viri replicate by copying their code into a program.
- This can mean overwriting the code that is there, or “patching” the virus code in, so that it gets executed when the program runs.
- A little care must be take to make sure that the code is run properly when the program executes, but basically, that is all there is to it.
- Note that this is essentially how programs are “patched” when the vendor finds a bug or problem with the code.

Patching

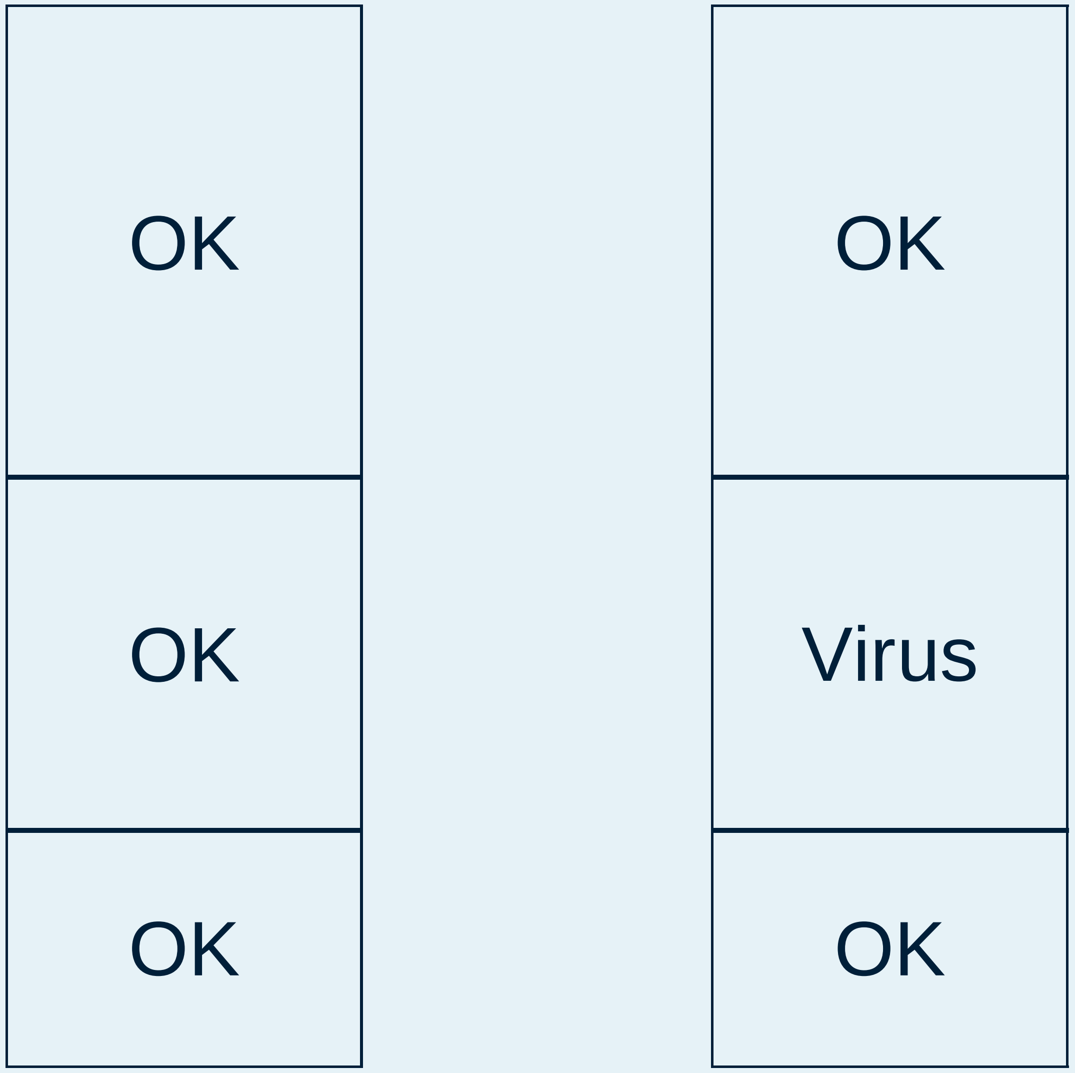


Virus as Patch



< > - +

Virus the Crude Way



Stealth

- Ideally the virus should go undetected.
- This means several things:
  - The size of the file and any time stamps should not change.
  - If possible, the virus action should be low profile, until it’s too late.
  - The infected program should appear to function normally.
- Not all viri care to this extent.

Detecting Viri

- Since the virus must place code in the program, the byte pattern of the code can be used to detect the virus.
- Some viri must be placed in particular parts of the disk or program, such as the boot sector, and so this reduces the amount that must be scanned.
- Thus (most) virus detection software is purely signature based. It looks for specific byte patterns that match a particular virus, or match a particular “bad action” that viri take.

Problems

- Signature based methods have false alarms.
- I am unaware of any studies that measure or characterize the false alarms of any commercial virus detection software.
- They do happen, though (I have had false alarms on gzipped tar balls of source code from a Linux box).
- Signatures also only work on the viri you know.

More Problems

- Virus writers can obtain the same anti-virus programs you can, and can ensure that theirs isn’t detected.
- They can make the virus modify itself as it propagates. These are called polymorphic viruses.
- They can change where the virus installs itself.
- They can have the virus encrypt itself, self extract, and play various similar games.

Other Approaches

- One way to detect viri is to run the program in a protected environment.
- This assumes the virus can’t tell it’s in a protected environment, or that the check can be detected.
- This can’t detect a virus set to “go off” at a particular time or after a particular event (unless you run in a protected environment all the time).
- This can be a pain to do for every single file you want to check.

Other Approaches

- Virus infections can be detected if you had a checksum (say, via tripwire) of the original uninfected program.
- This doesn’t work with email, unless some kind of encrypted authentication (digital signatures) is implemented.
- Like everything in security, digitally signing every piece of email is a pain, and people won’t do it until it is done for them.

Other Approaches

- Not to be a pessimist or anything, but having email programs do things for us is how we got into this mess in the first place.

< > - +

Other Approaches

- The final (automatic) approach to virus detection is to watch for “bad things”.
- Programs that act in a manner contrary to security policy are, by definition, “bad”.
- This includes writing places they shouldn’t, accessing programs, ports, etc that they shouldn’t. It can also mean using statistical methods of characterizing “normal” behavior.

Other Approaches

- One downside to this is that it can be “detection after the fact”. Still, it can provide the possibility of stopping the infection before it gets too widespread.

< > - +

Knowledge is Power

- Some kinds of viruses (email viruses, for example) can be stopped by word-of-mouth.
- If you get such a virus, announce it to all your friends, describing how they can detect the virus (e.g. the email subject).
- Know what your email reader does for you (automatically open word documents, etc.). Don’t let it do things without you’re ok.
- Keep abreast of the latest news on the newest viri and keep your software up to date (patches, virus signatures, etc.).

No Silver Bullet

- Fred Cohen gave the following proof that no perfect virus detector can exist:
  - Let D be a perfect virus detector: For any program P:
    - $D(P) = T$ if P is a virus.
    - $D(P) = F$ if P is not a virus.
  - Define the program V as:
    - if($D(V) = F$) then infect, otherwise do nothing.
  - If $D(V) = F$ then V is a virus, if $D(V) = T$ then it isn't.

< > - +

Silver Bullets

- If the virus detector says V is a virus then it isn't.
- If it says V isn't a virus, it is.
- Note that from the practical standpoint, this isn't a particularly useful counter-example:
    - The only time a virus is missed is when it isn't one.

< > - +

Silver Bullets

- However, this presupposes that the perfect virus detector can be built, which the counter-example shows is impossible. So, since D cannot be perfect, we can't be sure that the only time it makes a mistake is with V.

< > - +

Philosophical Interlude

- Defense is usually harder than offense. The “offender” knows the target, while the defender must defend against an unknown attack.
- Homogeneity in operating systems or applications is bad. Death by monoculture.
- Diversity is good in defenses.
- Diversity is bad, locally, in applications. It is good globally.

Philosophical Interlude

- Very secure systems are not as easy to use, or as “fun”, as unsecure ones. Thus, we will never have very secure systems for the masses.
- Your father/grandmother/Uncle Ernie (whoever you think is the most computer-illiterate person you know) is going to have a computer.

< > - +

Epidemiology

- The name “virus” obviously comes from biology.
- It makes sense to see what other concepts from biology can be used to study computer viruses.
- Since a virus spreads from computer to computer, in much the same way a disease spreads from person to person, it makes sense to analyze the spread of computer viruses using epidemiological models.

Terminology

- A computer is **infected** if the virus exists on the computer (in a manner such that the virus can be run or passed to another computer).
- A computer is **susceptible** to a virus if it could become infected with the virus, provided the virus is somehow introduced to the computer.
- Two computers have bf adequate contact if one would have transmitted a virus to the other had it been infected and had the other been susceptible.

< > - +

Terminology

- The **birth rate** of a virus is the frequency with which adequate contact occurs.
- A computer is **cured** of a virus if all copies of the virus are removed from the computer.
- The **death rate** of a virus is the frequency of cure.
- An **epidemic** is the widespread occurrence of a disease.

Terminology

- The **epidemic threshold** is the relationship between the birth rate and the death rate at which the virus becomes widespread.
- A virus is **extinct** if it can no longer infect any computer.
- The **extinction rate** is defined to be the ratio of the death rate to the birth rate.
- A disease that can maintain an epidemic for a long time is called **endemic**. For example, common childhood diseases are endemic.

< > - +

SIS Model

- Susceptible-Infected-Susceptible.
- All susceptibles are equally susceptible.
- Probability of any one susceptible being infected is a function only of the number of infected.
- Probability of an infected being cured (and hence becoming susceptible) is constant.

Deterministic SIS Model

$$\begin{tikzcd} \boxed{S} \arrow[r, "\beta I S"] & \boxed{I} \arrow[l, bend left, "\delta I"] \end{tikzcd}$$

- β is the infection rate.
- δ is the cure rate.
- Setting $N = S + I$, this results in the differential equation:
- $\frac{dI}{dt} = \beta I(N - I) - \delta I.$

Solution

- $\frac{dI}{dt} = \beta I(N - I) - \delta I.$
- Solution:
  - $I = \left(N - \frac{\delta}{\beta}\right) / \left(1 + Ce^{-(\beta N - \delta)t}\right)$
- C is a constant, depending on the number of infected computers.
- As $t \to \infty$, $I \to N - \frac{\delta}{\beta}$.

< > - +

Solution

- Note also that for very large N and fixed δ, β we obtain approximately the same value for I for all time.
- Recall that this is really a discrete system. Thus, if the ratio is less than 1, the virus goes extinct.

Stochastic vs Deterministic

- In real epidemics, there is always the possibility that everyone just happens to get cured all at once and the virus goes extinct.
- This is not modeled by the deterministic model.
- There is a certain amount of fluctuation in a real epidemic, one cannot say that at time t there will be I infected exactly.
- One needs to model the spread as a stochastic system rather than a deterministic one.

< > - +

Kephart and White

- Kephart and White construct a stochastic differential equation for the SIS model.
- This has the following form:
  - $\frac{dP(I,t)}{dt} = -a_I P(I,t) + b_I P(I_+,t) + c_I P(I_-,t)$
  - where $P(I,t)$ is the probability of I infected computers at time t, $I_+ = I + 1$ and $I_- = I - 1$.
- This results in a tri-diagonal set of coupled linear differential equations, which can be easily solved.

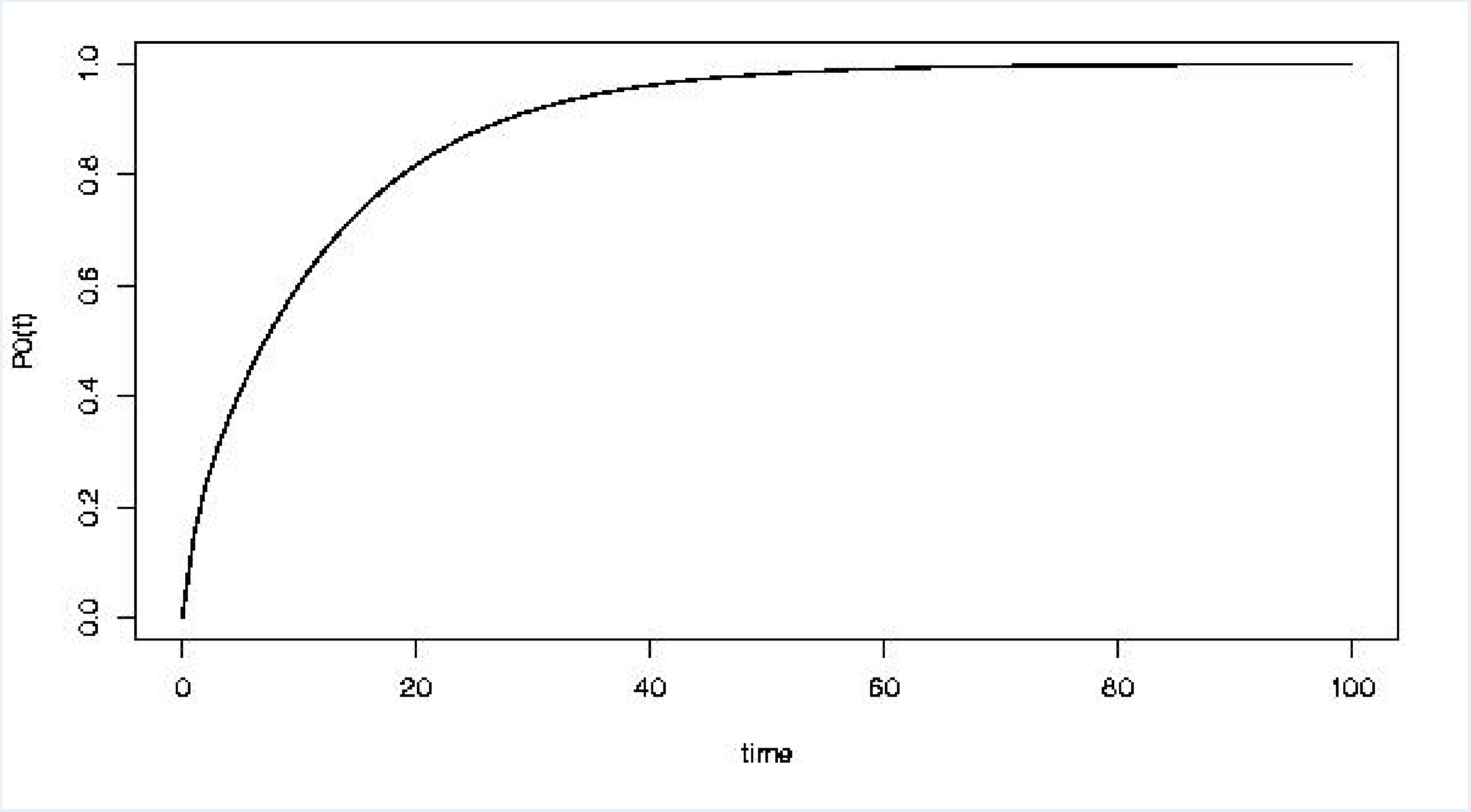
< > - +

Kephart and White Solution

- Writing $\mathbf{P}' = \mathbf{A}\mathbf{P}$ we have:
  - $\mathbf{P} = \alpha_0 \mathbf{x}^{(\mathbf{0})} e^{\lambda_0 t} + \ldots + \alpha_N \mathbf{x}^{(\mathbf{N})} e^{\lambda_N t}$
  - where the $\lambda_i, \mathbf{x}^{(\mathbf{i})}$ are the eigenvalues and eigenvectors of $\mathbf{A}$.
- Since $\mathbf{A}$ is tridiagonaly, this eigensystem is easy to obtain.

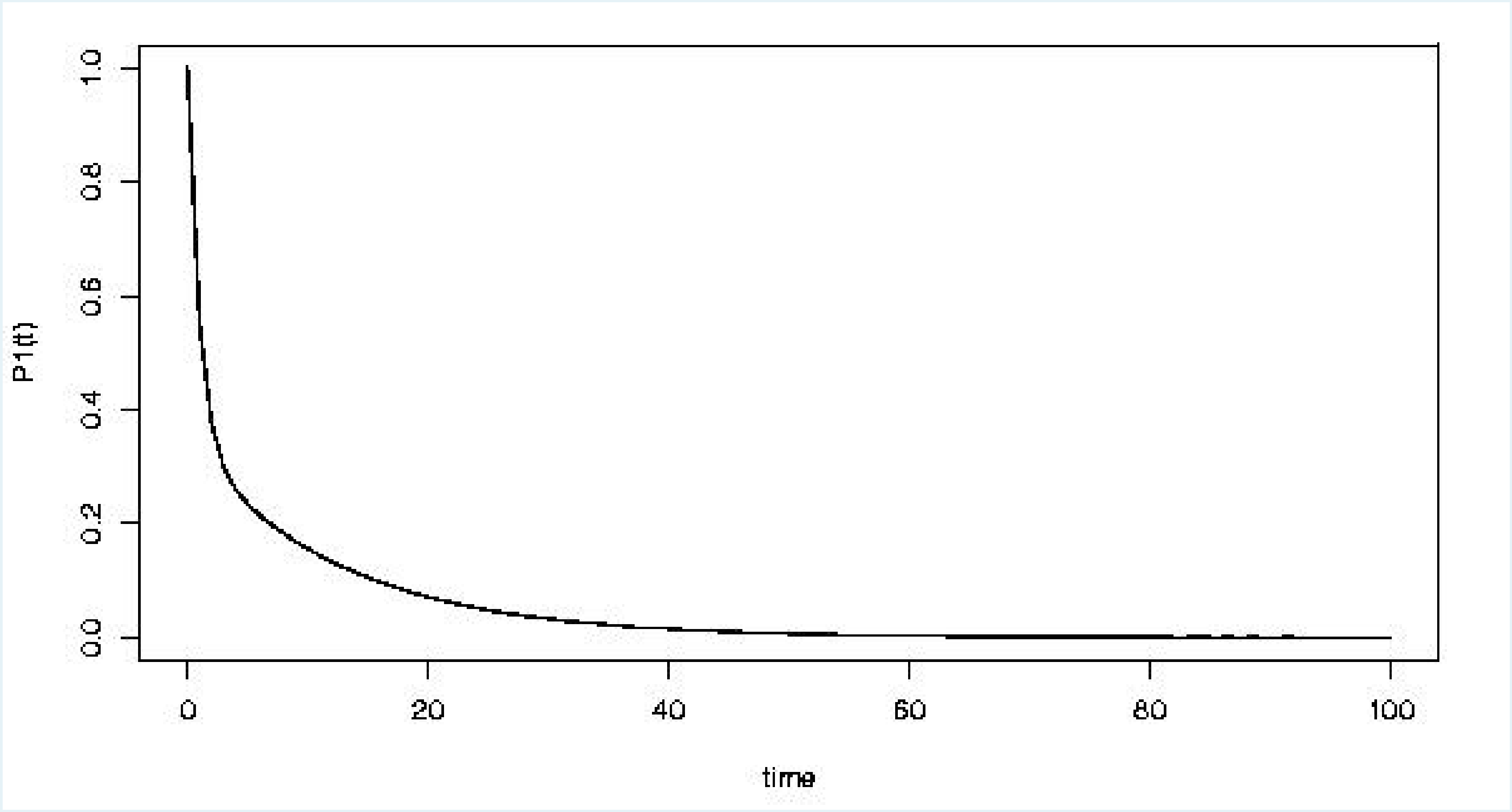
< > - +

Example: $N = 2, P_0$



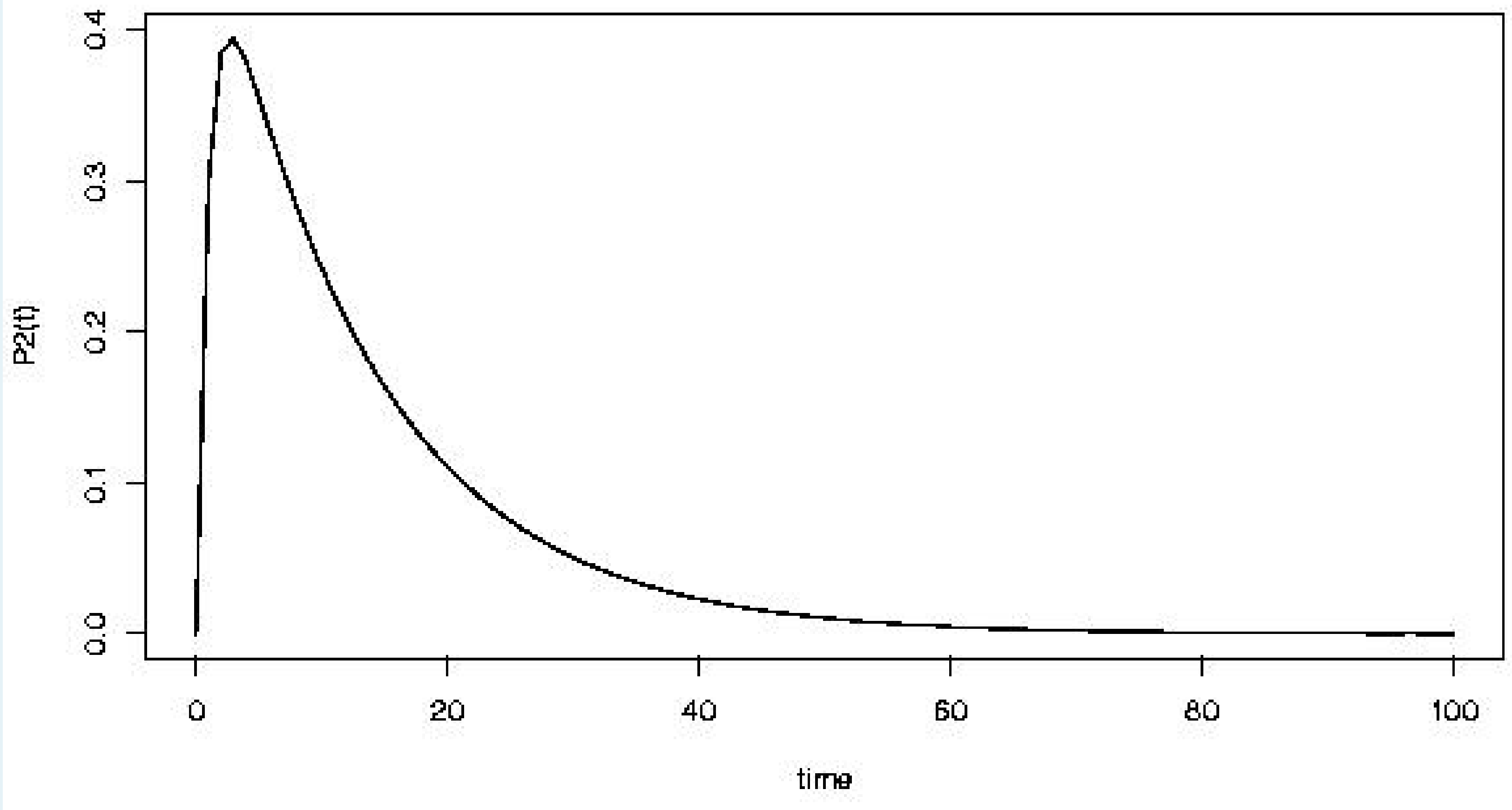
< > - +

Example: $N = 2, P_1$



< > - +

Example: $N = 2, P_2$



< > - +

SIS with Reintroduction

- Susceptible-Infected-Susceptible.
- All susceptibles are equally susceptible.
- Probability of any one susceptible being infected is a function only of the number of infected.
- Probability of an infected being cured (and hence becoming susceptible) is constant.
- If the number of infecteds is zero, there is a fixed probability of reinfection (reintroduction of the virus).

Birth and Death

- Model the SIS model as an $n+1$ state continuous time Markov process, where the states denote the number of infected machines.
- Represent the process as a birth-and-death process with birth rates

$$\lambda_i = ri(n-i)$$

and death rates

$$\mu_i = ci,$$

Birth and Death

- c is the cure rate for a single infected computer, and r is infection rate from one infected computer to one susceptible computer.

Solutions

- The form of the stationary distribution is given by the formula:

$$P_0 = \frac{1}{1 + \sum_{i=1}^{n} \frac{\lambda_0 \lambda_1 \cdots \lambda_{i-1}}{\mu_1 \mu_2 \cdots \mu_i}}$$

$$P_k = P_0 \frac{\lambda_0 \lambda_1 \cdots \lambda_{k-1}}{\mu_1 \mu_2 \cdots \mu_k}.$$

- where

$$\frac{\lambda_0 \lambda_1 \cdots \lambda_{k-1}}{\mu_1 \mu_2 \cdots \mu_k} = \frac{a r^{k-1} (n-1)!}{k c^k (n-k)!}.$$

Mathematica!

Mathematica to the rescue:

$$
\begin{aligned}
P_0 &= \frac{c}{c + a\ {}_pF_q[\{1,1,n-1\},\{2\},-\frac{r}{c}]},\\
P_k &= \frac{ar^{k-1}(n-1)!}{c^{k-1}k(n-k)!(c + a\ {}_pF_q[\{1,1,n-1\},2,-\frac{r}{c}])}.
\end{aligned}
$$

These can be used to compute various things like the mean, mode, etc.

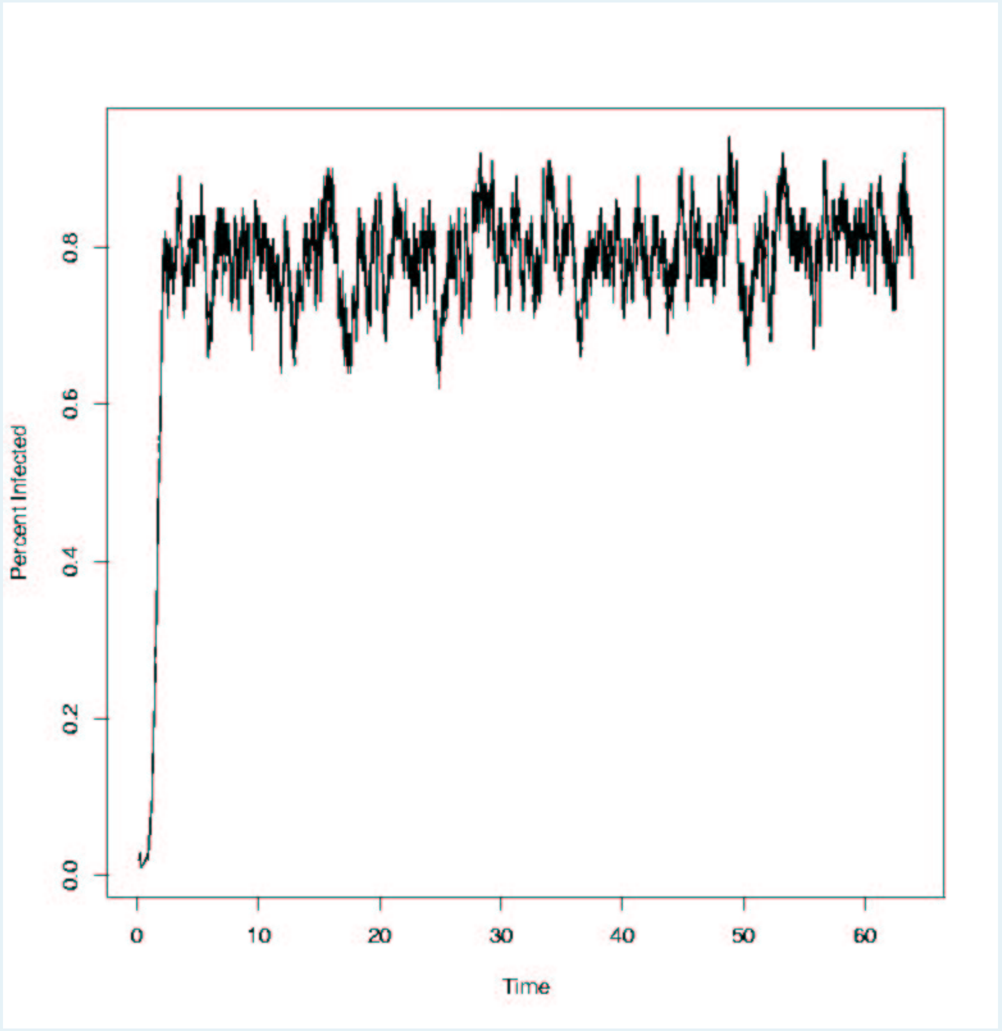
< > - +

Results

- Various asymptotic results are available.
- The reintroduction allows us to treat this as an approximation to the quasi-stationary regime.
- The properties of the epidemic depend on c/r.
- Small values result in a Poisson distribution.
- Moderate values are asymptotically normal.
- Large values lead to a logarithmic limit.

< > - +

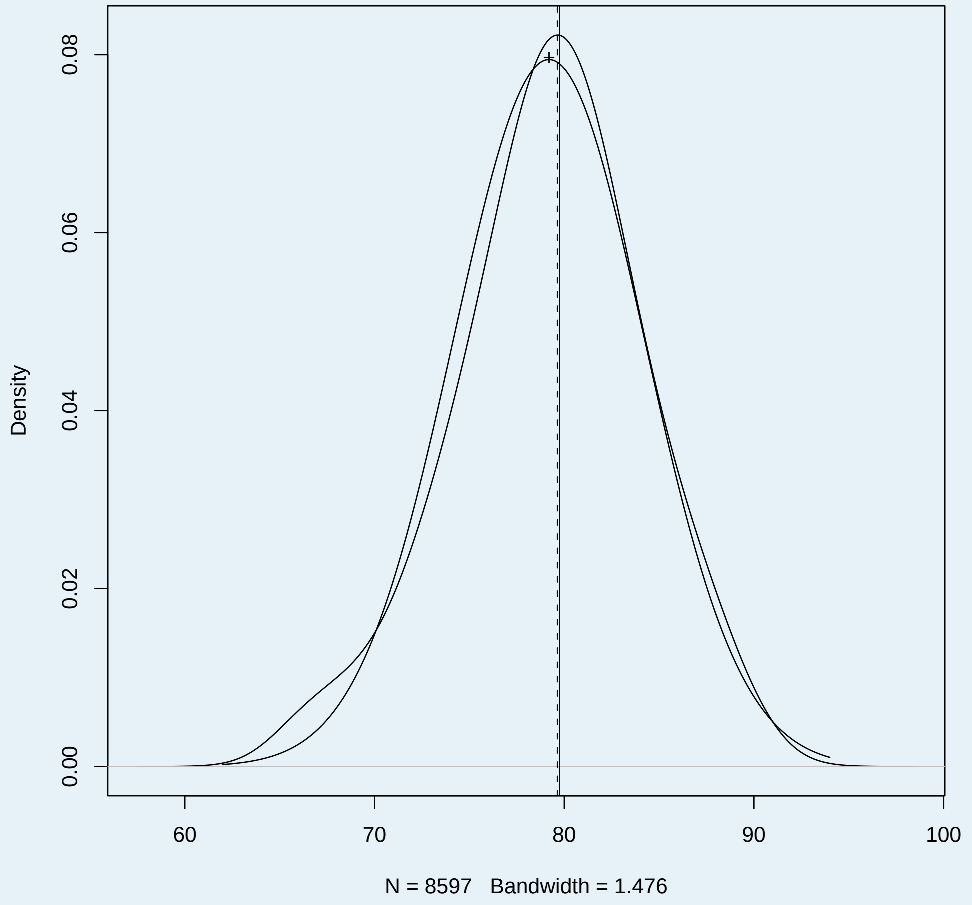
A typical simulation



$a = 1$, $r = 0.05$, $c = 1$ and $n = 100$.

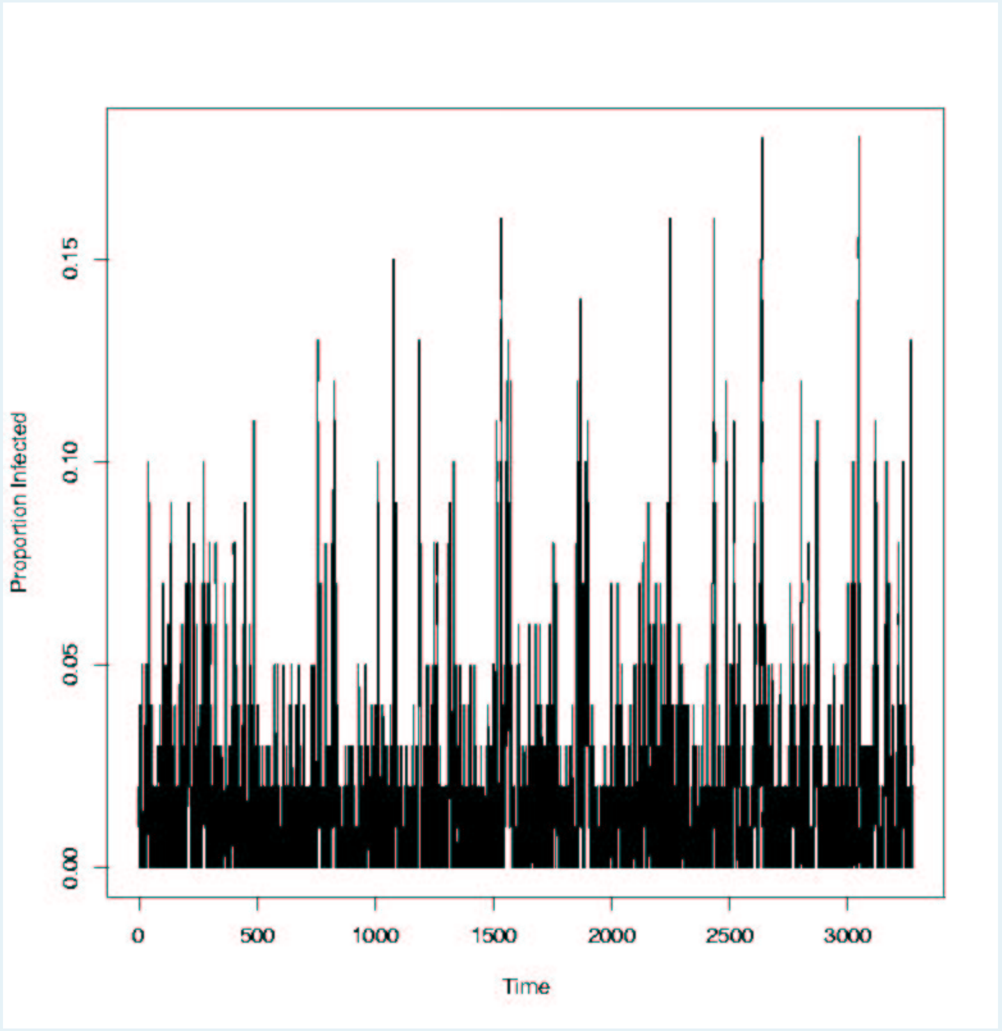
< > - +

Density



$a = 1$, $r = 0.05$, $c = 1$ and $n = 100$.

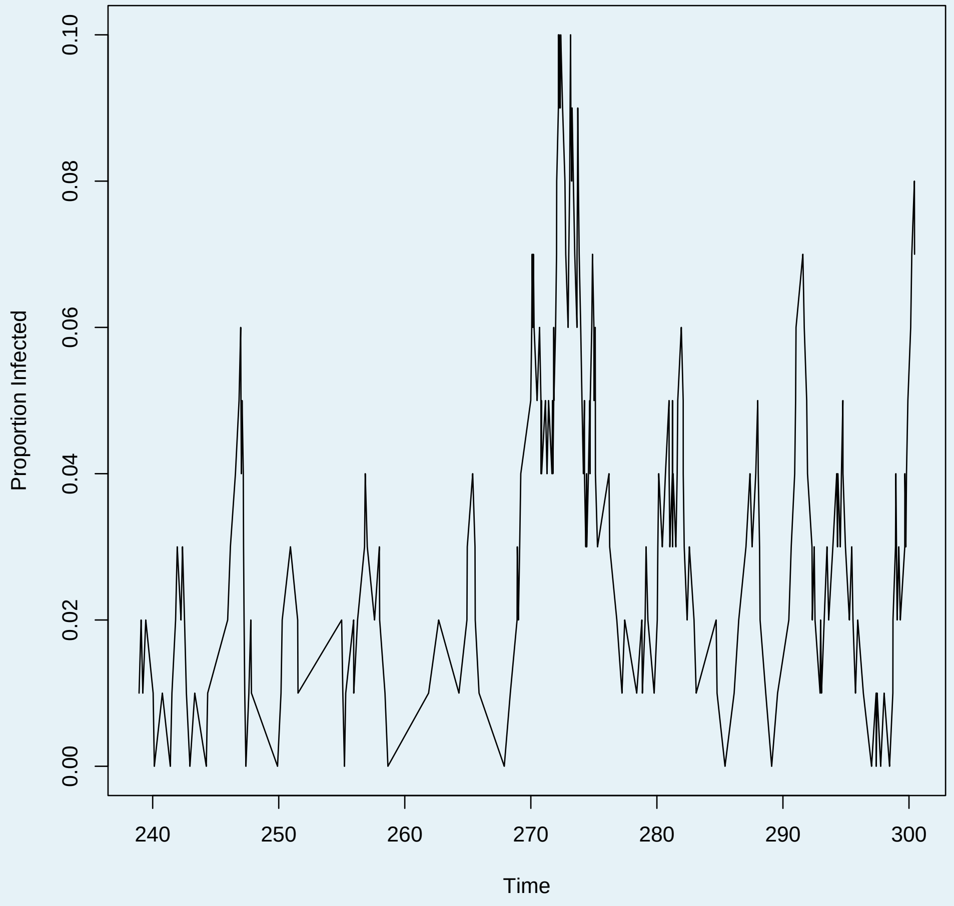
Another simulation



$a = 1$, $r = 0.008$, $c = 1$ and $n = 100$.

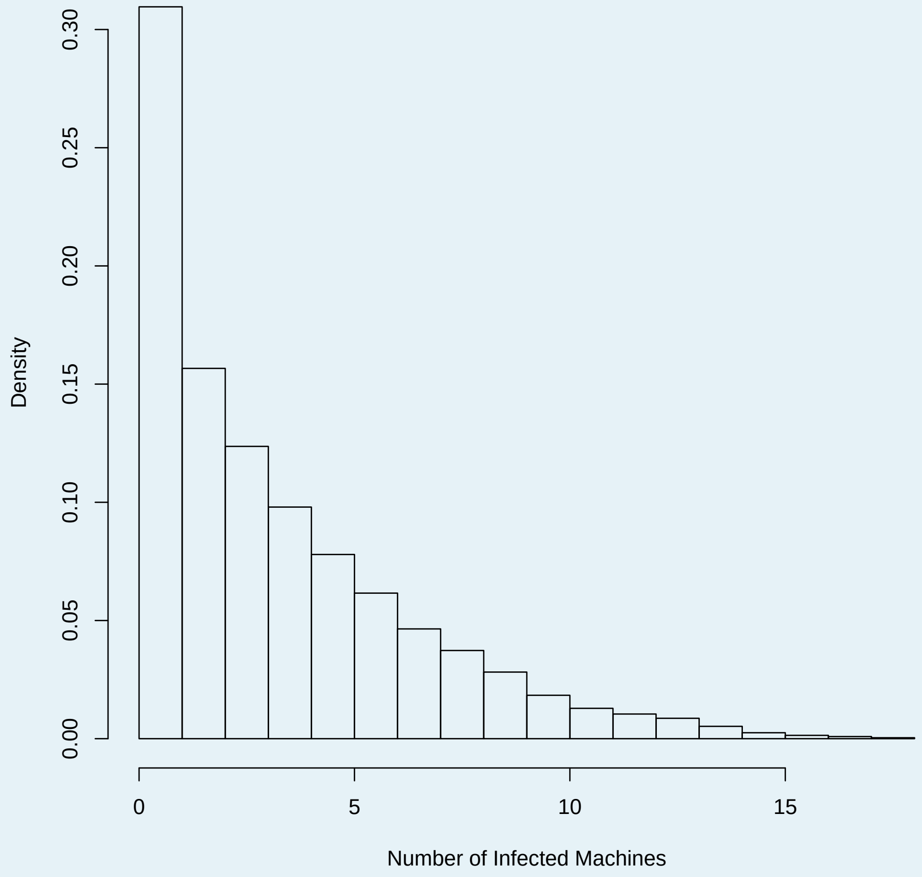
< > - +

Zoom



$a = 1$, $r = 0.008$, $c = 1$ and $n = 100$.

Density



$a = 1$, $r = 0.008$, $c = 1$ and $n = 100$.

< > - +

Other Extensions

- Different types of computers (different rates of cure and infection).
- Nonhomogeneity (some computers interact more than others).
- Preventative action (anti-virus software).
- Computers become less susceptible after an infection.
- Computer susceptibility changes with time (anti-virus signatures are not kept up-to-date, etc.)

Immunology

- As mentioned, one problem with virus scanners is that they need a pattern to match against.
- The analogy with immunology is that one wants “antigens” that are coded to detect “bad” code.
- One way of thinking about this is that the antigens detect “non-self”.
- This analogy is the basis for the modification of the n-gram approach to virus detection.

n-gram Antigens

- Think of the signatures as n-grams designed to detect a specific pattern.
- This produces an n-gram approach where n is variable across the different signatures. So, in this case, the n-gram is the pattern of n bytes that characterize the signature for the virus.

n-gram Antigens

- The next extension is to generate n-grams designed to detect viruses that have not yet been detected. How to do this?
- Generate n-grams at random. Check these against programs that are known to be uninfected. Those n-grams that match “good” code (self) are thrown out. The ones that remain are considered to be examples of “bad” code.

n-gram Antigens

- Some modifications might be in order.
- For example, one might start with a set of virus signatures and mutate them (slight modifications to the byte pattern), rather than purely at random.
- Or, one may start with a “good” patterns (patterns taken from clean files) and mutate them.
- One may also take “bad things” (writing to the boot sector) and put them together to make potential virus signatures.

Virus Honeypots

- Another way to get virus signatures is to become infected with a lot of viruses.
- This may not be as dumb an idea as one might at first think. The idea is to have a virus honeypot.
- The idea of a virus honeypot is to have a file or files that are highly likely to be infected by viruses. Monitor these files for infection.
- Ideally, the entire machine involved in this experiment is an expendable machine whose only purpose is to be infected by viri.

Worms

- A worm is a program that spawns copies of itself. Or, if you prefer, a program that copies itself without human intervention.
- This can be on a single machine, in which case the worm can use up resources.
- Or it can be across a network, where the worm copies itself to other machines.
- Besides using up resources, worms, like viruses, often perform various other (generally bad) functions, such as obtaining information or destroying files.

< > - +

The Internet Worm

- November 2, 1988. A worm was released onto the Internet (such as it was at the time) and quickly reproduced.
- The Internet worm was not meant to be malicious, apparently, but an error in the code made it reproduce far faster than intended.
- The Internet worm may have infected as much as 10% of the machines on the Internet.

The Internet Worm

- The worm spread from machine to machine using several exploits. It was not supposed to run more than one copy on a machine, and basically it’s purpose was to try to infect as many machines as possible, basically to show it could.

The Internet Worm

- The Internet worm used several methods to gain access.
  - It used an exploit against the sendmail program.
  - It used an exploit against the finger program.
  - It cracked passwords and tried to log on as users whose passwords it had cracked.

< > - +

The Internet Worm

- Once it obtained access, it sent a bootstrap program, compiled it, then initiated a connection to bring over the rest of the code.
- Once running, it removed its files from the disk and changed its process name to look innocuous.

Consequences

- Many machines were brought down completely.
- Several institutions (including the one where I worked at the time) were taken completely off the network by the worm.
- The cost has probably never been tabulated, and would probably be meaningless.

Consequences

- The number of machines on the Internet at the time was in the tens of thousands, and the worm could be removed from an infected computer by a simple reboot (although more needed to be done to keep it from being reinfected).
- The guy who wrote the worm was ruined (and arrested).

Macro Worms

- Macro worms are programs sent through email, that exploit a “feature” that allows programs to be run when certain Microsoft documents are opened.
- By the “no human intervention” definition, these are viruses, rather than worms. I’ll call them worms.
- Macro worms typically reproduce by emailing themselves to your friends.

Macro Worms

- The beauty of this scheme was that the virus always came from someone you knew, rather than some stranger spamming you.
- This meant that they spread very quickly.

< > - +

Melissa

- The most famous, and first, of the macro worms was Melissa.
- The virus came as an email from someone you knew.
- It had a document attached, with the message “Here is the document you asked for”.
- When the document was opened, a macro program was run that did a number of interesting and nasty things.

Melissa

- Note that many mailers (helpful little critters that they are) automatically would open the attachment for you, thereby causing the virus to be run simply by your act of reading your email.

< > - +

Melissa

- Melissa performed several actions when the email was read:
  - First, it sent itself to the first 50 addresses in your address book.
  - It infected your Word software so that new documents you created would be infected.
  - It changed the security settings on your Word program so that your system was more vulnerable, and harder to make invulnerable.

Melissa

- Melissa cost millions of dollars (so they say). It did cause many places to lose Internet access, sometimes for more than a day.

Melissa Upsides

- One upside of Melissa is that it made people more aware of the security holes in certain software systems. If you call that an upside.
- This has created quite a lot of discussion about just how helpful these applications should be.
- This is a very good thing. There will always be a tension between security and ease of use.

More Melissa Upsides

- Another upside is it told us something interesting about MS Word:
- MS Word places information about the person who generated the document (or more properly the person who registered Word) at the end of each document. This was how the writer of Melissa was caught.

More Melissa Upsides

- Another useful tidbit that you should know is that MS Word documents come in fixed size increments. What this means to you is that if you send a document of a length between these increments it gets padded out by whatever happens to be on the disk after the document.

I Love You

- So called because the subject was “I love you” (who could resist opening such a missive from a close friend?), the I Love You virus was another macro worm.
- It was written to gain access to computer accounts without having to pay for them.
- It sent account and password information to the author.
- Like Melissa, it spread by sending email to people in your address book.

I Love You

- It also changed your default home page on your browser to a site that would execute the virus.
- It also modified sever types of files to execute the virus. In this sense it really was a true virus, as well as a worm.

< > - +

Warning: Soapbox Alert

- Windows is an icon-driven operating system.
- Everything is supposed to be accessed through “clicking” on icons.
- Like it’s predecessor, MSDOS, Windows still uses the extension (the thing after the dot) to tell what kind of program should be executed to process any given file.
- This is a really dumb idea.

Warning: Soapbox Alert

- Furthermore, the default for file browsers on Windows is to not let the user worry his or her pretty little head about these file extensions, and not show them.
- This is a particularly dumb idea.
- There is a new attack that takes advantage of this.

< > - +

ILY

- By changing image and sound files into ones that execute the virus, ILY became very hard to clean off an infected computer.
- An operating system that made you type:

  playwav wavfile

  to listen to a sound file could not be tricked in this manner.
- By changing the extension (and putting its code in the file) ILY was able to turn any file into a program that spawned a copy of the virus.

Philosophical Interlude

- Email readers should not do anything beyond display.
- Documents should not execute code, without the user explicitly allowing it (this is not enough, but at least it's a start).
- Mobile code is a very cool idea that only works in a trusted environment.
- There are very few trusted environments on the Internet.

Philosophical Interlude

- Still, mobile code should always announce itself, and allow the user to decide if they want some unknown code to run on their computer.
- Perpetual prediction: the worst is yet to come.

Ramen

- Worms are not just for Windows.
- Ramen was a worm that infected Linux machines.
- It utilized a suite of attack tools to compromise a computer.
- Once it found a vulnerable computer, it loaded itself on the computer and opened up a port (27374) which allows anyone connecting to the port to obtain a copy of the virus.

Ramen

- It sends email announcing the compromise.
- It then scans for new machines to infect.
- Note: Things like port 27374 are trivial to change, so don’t rely on them too heavily to detect any specific attack tool.

Detecting Worms

- Look for unusual activity:
  - Connections/probes from other machines.
  - Unusual files appearing on your disk.
  - Changes to existing files.
  - Strange programs running.
  - Load averages too high.
  - Outgoing connections.
- Use many different monitoring programs to try to detect weirdness before it's too late.

< > - +

Defending Against Worms

- Don’t let applications execute programs unless you tell them to.
- Don’t execute programs unless you think you know what they are supposed to do.
- Patch.

Further Reading

- Cohen, “Computer Viruses, Theory and Experiments”, Computers and Security, 6, 1987, 22–35.
- Ashmanov and Kasperskaya, “The Virus Encyclopedia: Reaching a New Level of Information Comfort”, IEEE Multimedia, 6, 1999, 81–84.
- Denning, *Computers Under Attack: Intruders, Worms, and Viruses*, 1990, Addison-Wesley.

< > - +

Further Reading

- McAfee and Haynes, *Computer Viruses, Worms, Data Diddlers, Killer Programs, and Other Threats to Your System*, St. Martin's Press, 1989.
- Kephart and White, "Directed-Graph Epidemiological Models of Computer Viruses", Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, 1991, 343-359.
- Kephart and White, "Computers and Epidemiology", IEEE Spectrum, 30, 20–26,1993.

< > - +

Further Reading

- Marmelstein et al., “A Distributed Architecture of an Adaptive Computer Virus Immune System”, 1998 IEEE International Conference on Systems, Man, and Cybernetics, 3838–3843.
- Garber, “Melissa Virus Creates a New Type of Threat”, Computer, 32, 1999, 16–19.
- Nachenberg, “Computer Virus-Antivirus Coevolution”, Communications of the ACM, 40, 1997, 46–51.

< > - +

Further Reading

- Andersson and Britton, *Epidemic Models and Their Statistical Analysis*, Springer, 2000.
- Daley and Gani, *Epidemic Modelling: An Introduction*, Cambridge University Press, 2000.

< > - +