Network Forensics and Next Generation Internet Attacks

Moderated by: Moheeb Rajab

Background singers: Jay and Fabian

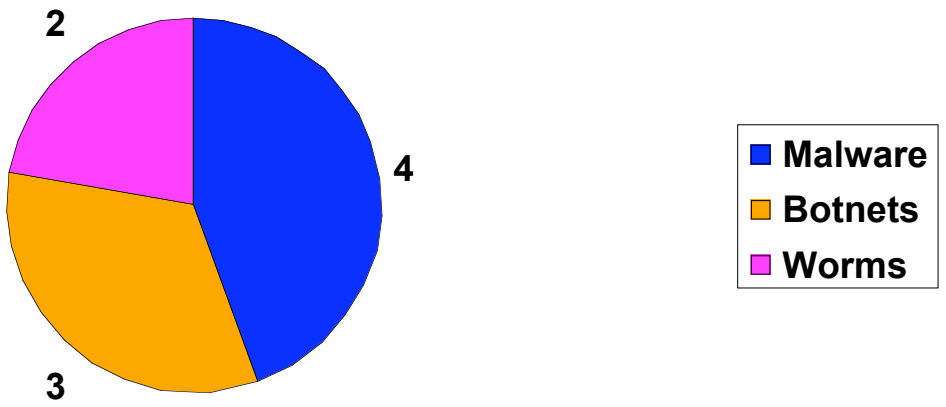
Agenda

- Questions and Critique of Timezones paper
  - Extensions
- Network Monitoring (recap)
- Post-Mortem Analysis
  - Background and Realms
  - Problem of Identifying Patient zero
  - Detecting Initial hit-list
- Next Generation attacks (Omitted from slides)
  - Implications and Challenges?

Botnets or Worms ?!

- *"The authors don't provide evidence that botnets propagate in the same way like regular worms"*

- Opening Sentence:

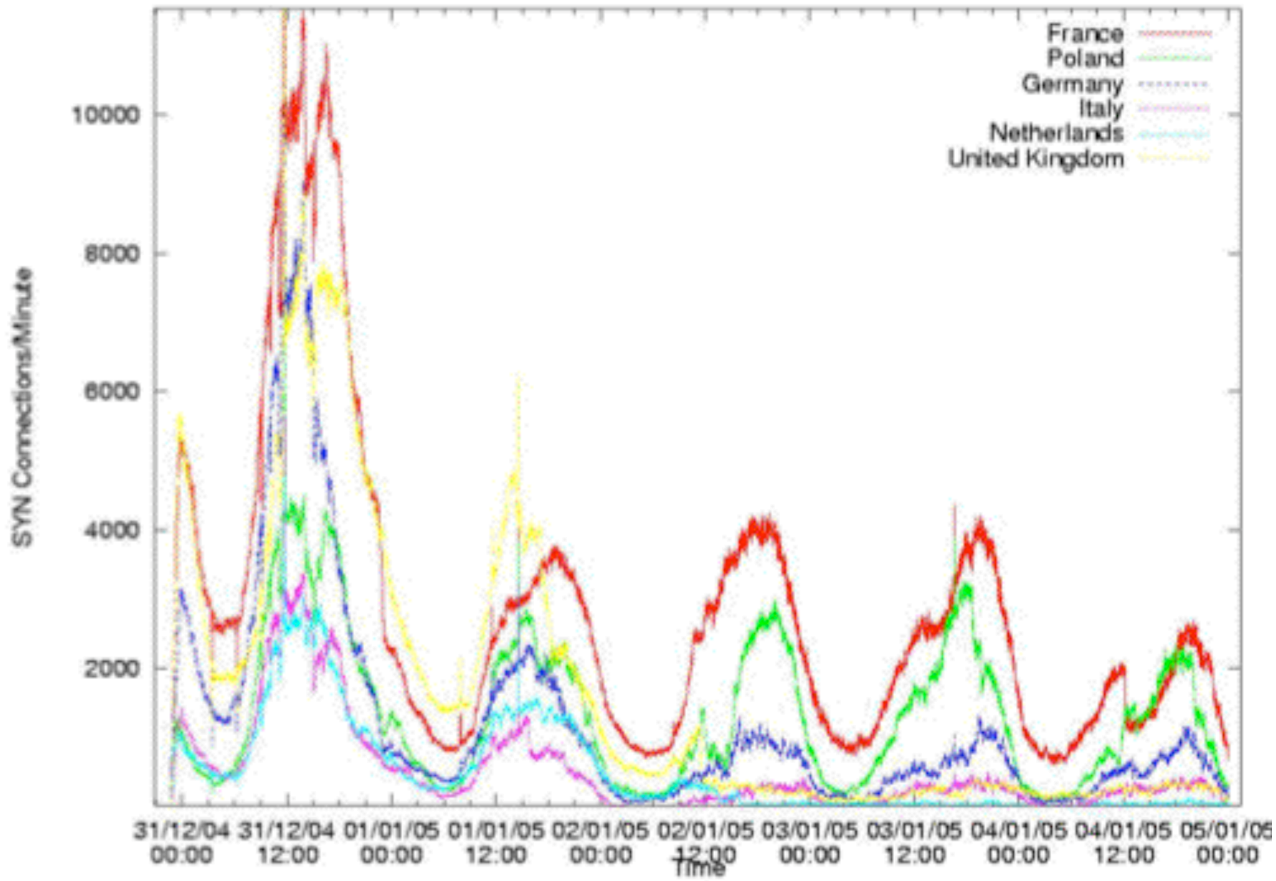


Student questions

Data Collection

- "*The original data collection method itself is worth mentioning as a strength of this paper*"
- "*Can't someone who sees all the traffic intended for a C&C server do more than simply gather SYN statistics*"
- "*It is not clear to me how do they know that they captured the propagation phase in their tests*"

Measuring Botnet Size



SYN Counting

- Only looking at the Transport Layer
  - Do we even know what this traffic is?
- DHCP'd hosts
  - DHCP will cause SYNs coming from different addresses.
- How does the Tarpit help?
- Totally unrelated traffic
  - Scans, exploit attempts, etc.

Estimating botnet size

- How do we quantify these effects and relate them back to the claimed 350 K size?
  - Are we counting wrong? If we assume DHCP lease of Δ hours, how do these projections change?
- Studied 50 botnets but we have 3 data points.
- Fitting the model to the collected data
  - What parameters did they use?

Evidence from “Da-list”

Date and Time	*DNS*	*Non-DNS*
Feb,1st 4:00 AM EST	49	4
Feb 1st 11:00 AM EST	23 (> 4 public IRCds)	4

General consensus

- Contrary to authors the attackers could use the timezones effect to their benefit
  - How?
- This is old-school, right?:
  - *Zhou et al*. A first look at P2P worms: Threats and Defenses. IPTPS, 2005.
  - Botnet Herders can hide behind VoIP. InfoWeek, 2/27/06
    - Okay, this is getting ridiculous
- Cherry-picking: some weird indications …

Extensions

- Can we use this idea for containment?
  - Query to know if someone is infected
  - How to preserve privacy and anonymity?
    - See *Privacy-Preserving Data Mining*. R. Agrawal and R. Srikant. *Proceedings of SIGMOD, 2000*
- Patching rates?
  - More grounded parameters might really affect model
  - How might we get this?
- Lifetime?

Student Extensions

- Is there better ways to track botnets other than poisoning DNS?
  - Crazy idea #1: Anti-worm
- Crazy idea #2: Statistical responders
  - Better way: *Weidong Cui et al*. **Protocol-Independent Adaptive Relay of Application Dialog**. In *NDSS 2006*
- What would you have liked to see with this data?

Using telescopes for network forensics

Forensic (Post-mortem) analysis

- Infer characteristics of the attack
  - Population size, demographics, distribution
  - Infection rate, scanning behavior .. etc
- Trace the attack back to its origin(s)
  - Identifying patient zero
  - Identifying the hit-list (if any)
  - Reconstructing the infection tree

Worm Evolution Tracking Realms

- Graph Reconstruction
- Reverse Engineering
- Timing Analysis

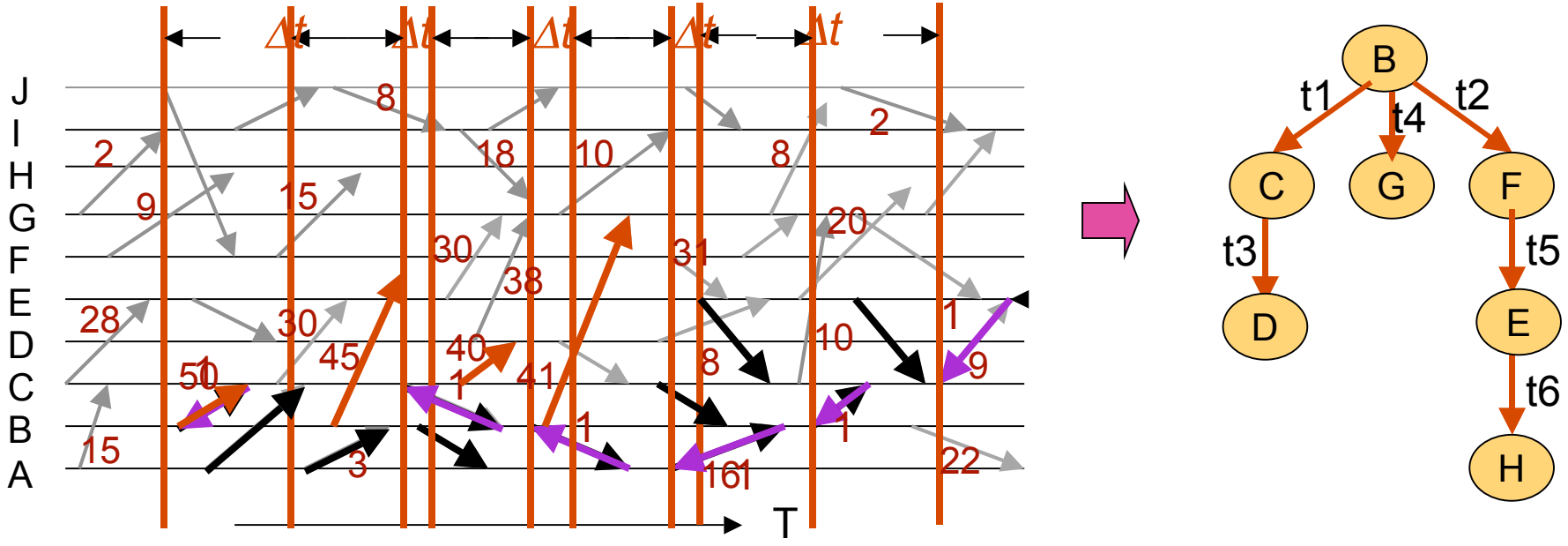
Infection Graph Reconstruction

Xie et al, "**Worm Origin Identification Using Random Moonwalks**" IEEE Symposium on Security and Privacy, 2005

- Proposed a random walk algorithm on the hosts contact graph
  - Provides who infected whom tree
  - Identifies the worm entry point(s) to a local network or administrative domain.

Random Moonwalks

- A random moonwalk on the host contact graph:
  - Start with an arbitrarily chosen flow
  - Pick a next step flow randomly to walk **backward in time**
- Observation: epidemic attacks have a **tree** structure
- **Initial causal flows emerge as high frequency flows**



Slide by: Ed Knightly

Random Moonwalk (Limitations)

- Host Contact graph is known.
  - requires extensive logging of host contacts throughout the network
- Only able to reconstruct infection history on a local scale
- *Careful* selection of parameters to guarantee the convergence of the algorithms
  - How to address this is left as open problem

Outwitting the Witty

Kumar et al, "**Exploiting Underlying Structure for Detailed Reconstruction of an Internet-scale Event**", IMC 2005

- Exploits the structure of the random number generator used by the worm
  - Careful analysis of the worm payload allows us to reconstruct the infection series

Witty Code !

```
srand(seed) { X ← seed }
rand() { X ← X*214013 + 2531011; return X }

main()
1.  srand(get_tick_count());
2.  for(i=0;i<20,000;i++)
3.      dest_ip ← rand()[0..15] || rand()[0..15]
4.      dest_port ← rand()[0..15]
5.      packetsize ← 768 + rand()[0..8]
6.      packetcontents ← top-of-stack
7.      sendto()
8.  if(open_physical_disk(rand()[13..15] ))
9.      write(rand()[0..14] || 0x4e20)
10.     goto 1
11. else goto 2
```

Witty Code!

- Each Witty packet makes 4 calls to `rand()`
- If first call to **rand**() returns X_i :

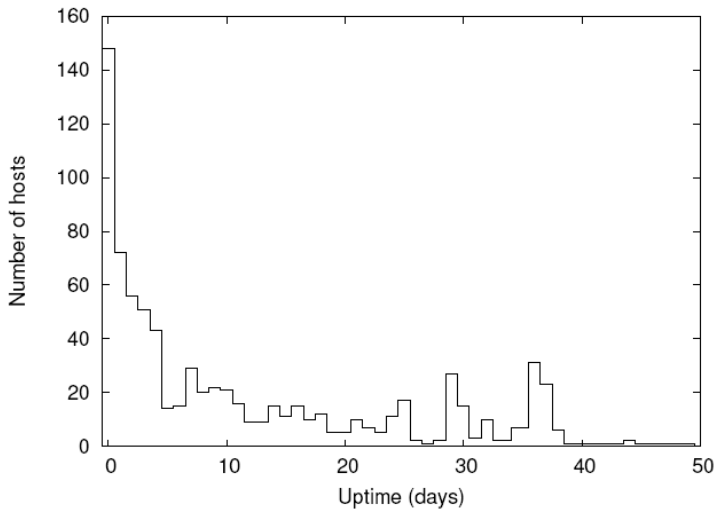
3. *dest_ip* ← $(X_i)_{[0..15]}$ || $(X_{I+1})_{[0..15]}$
4. *dest_port* ← $(X_{I+2})_{[0..15]}$

Given top 16 bits of X_i, now *brute force* all possible lower 16 bits to find which yield consistent top 16 bits for X_{I+1} & X_{I+2}

⇒ **Single** Witty packet suffices to extract infectee's *complete* PRNG state!

Interesting Observations

- Reveals interesting facts about 700 infected hosts:
  - Uptime of infected machines
    - Number of available disks
  - Bandwidth Connectivity
  - Who-infected whom
  - Existence of hit-list
  - Patient zero (?)



Reverse Engineering (Limitations)

- Not easily generalizable
  - Needs to be done on a case by case basis
- Can be tedious (go back to the paper to see).
- There must be an easier way, right?

Timing Analysis

Moheeb Rajab et al. **"Worm Evolution Tracking via Timing Analysis"**, ACM WORM 2005

- Uses blind analysis of inter-arrival times at a network telescope to infer the worm evolution.

Problem Statement and Goals

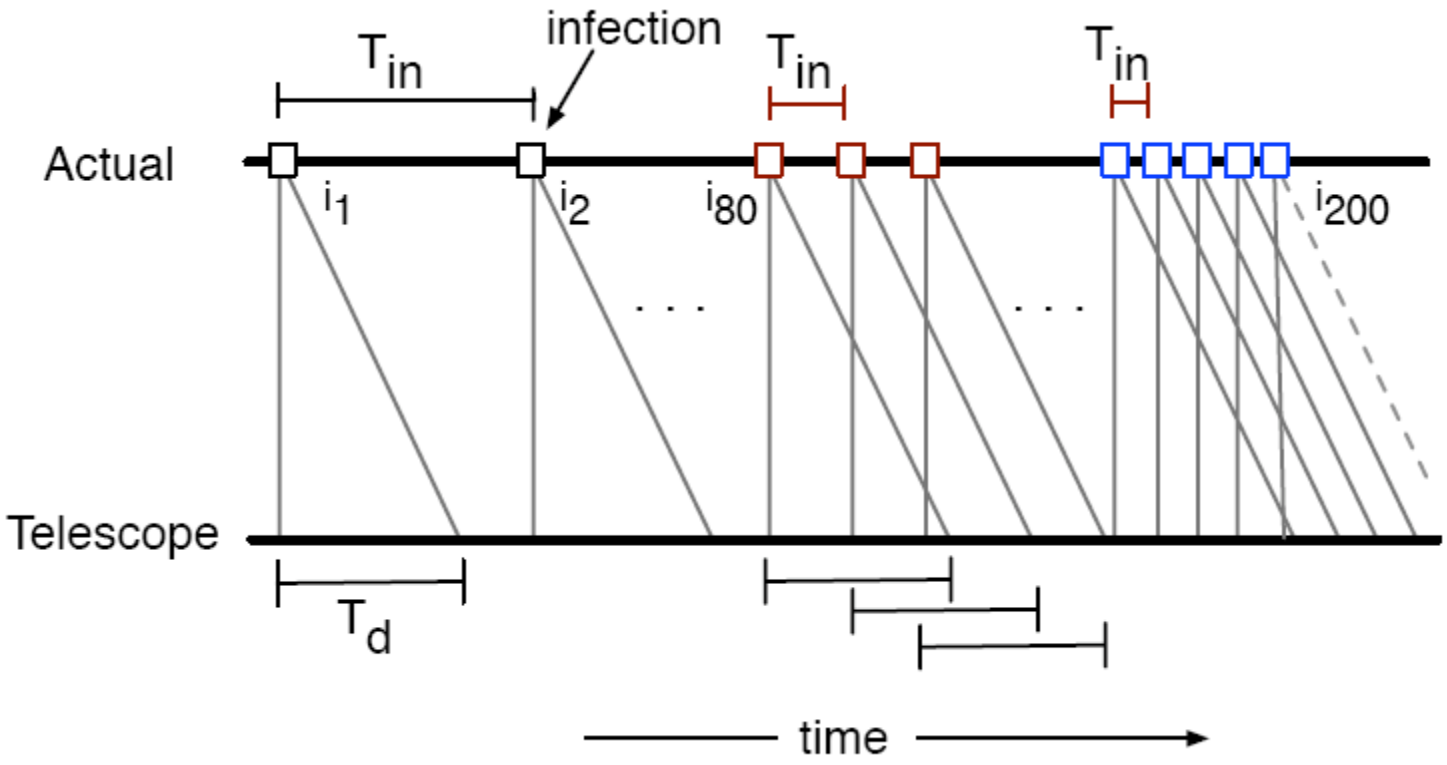
Consider a uniform scanning worm with scanning rate s and vulnerable population size V and a monitor with effective size M.

- To what extent can a network monitor trace the infection sequence back to patient zero by observing the order of unique source contacts?
- For worms that start with a hitlist, can we use network monitors to detect the existence of the hitlist and determine its size?

Evolution Sequence and "Patient Zero"

- We distinguish between two processes:
  - Time to Infect T_{in}
    - Time elapsed before the worm infects an additional host
  - Time to Detect T_d
    - The time interval within which a monitor can reliably detect at least one scan from a *single* newly infected host

Time to Infect *and* Time to Detect



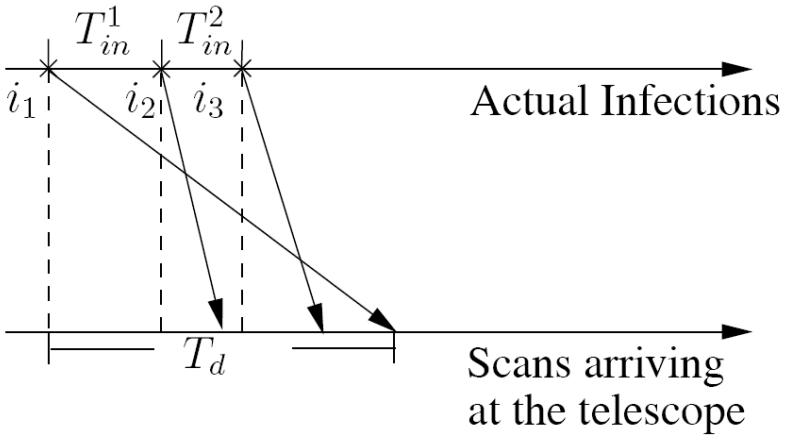
Time to Infect *and* Time to Detect

- Time to infect a new host T_{in}

$$T_{in} = \frac{\log\left(1 - \frac{1}{V - n_i}\right)}{sn_i \log\left(1 - \frac{1}{2^{32}}\right)}$$

Monitor Accuracy

- Monitor Detection time, T_d



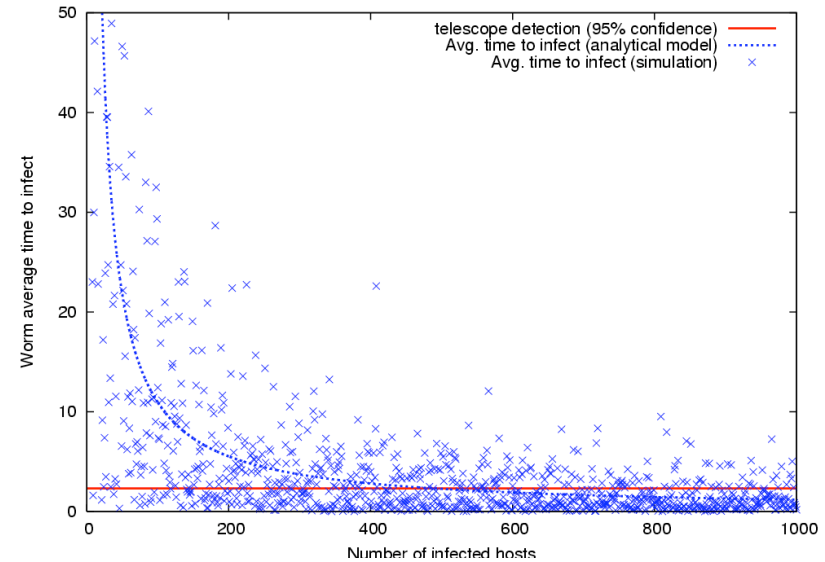
- Probability of error

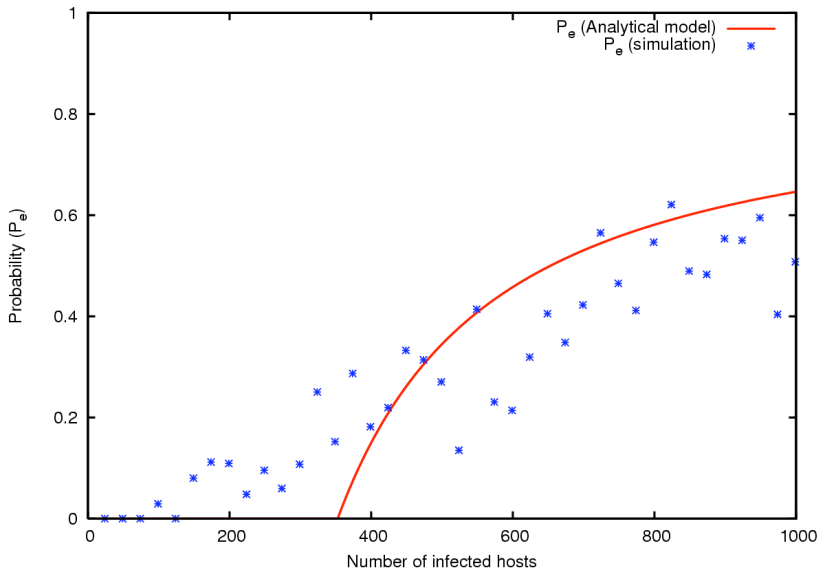
$$P_e = 1 - \prod_{i=1}^{n} \left(1 - \frac{M}{2^{32}}\right)^{\left(T_d - \sum_{j=1}^{i} T_{in}^j\right) s}$$

T_{in} and T_d

Uniform scanning worm:
s = 350 scans/sec,
V = 12,000
Monitor size = /8

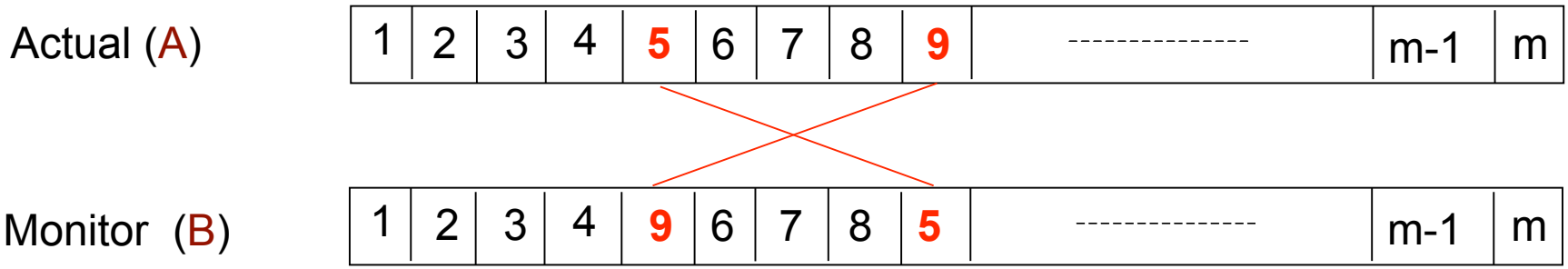
Probability of Error →





Infection Sequence Similarity

- Sequence Similarity

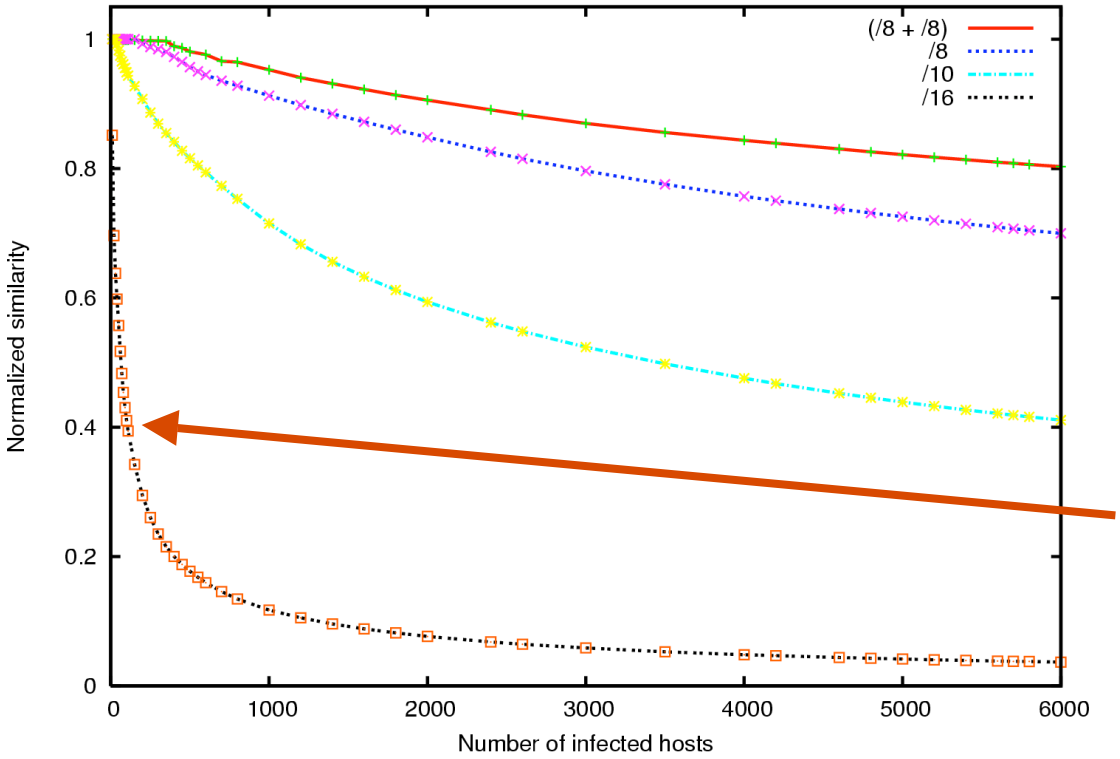


$$Y_{B \to A} = \sum_{i=0}^{m} \frac{\left(m - r_{(e_i, A)}\right)}{1 + \left| r_{(e_i, B)} - r_{(e_i, A)} \right|}$$

Is this any good?

- Two (interesting) cases:
  - Varying monitor sizes
  - Non-homogeneous scanning rates

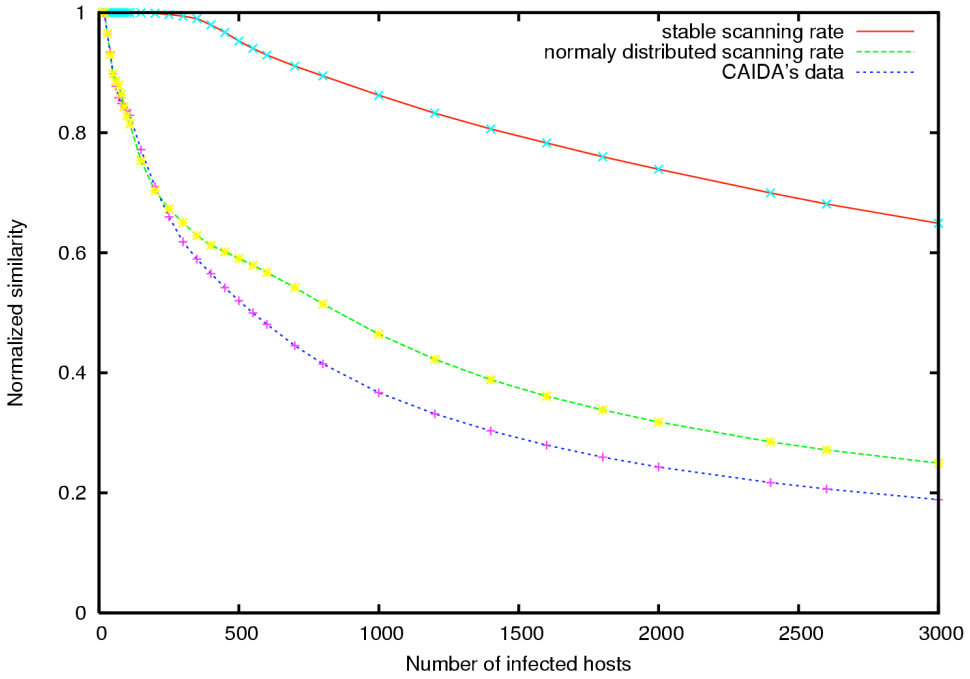
Bigger is Better



Larger telescopes provide a highly similar view to the actual worm evolution

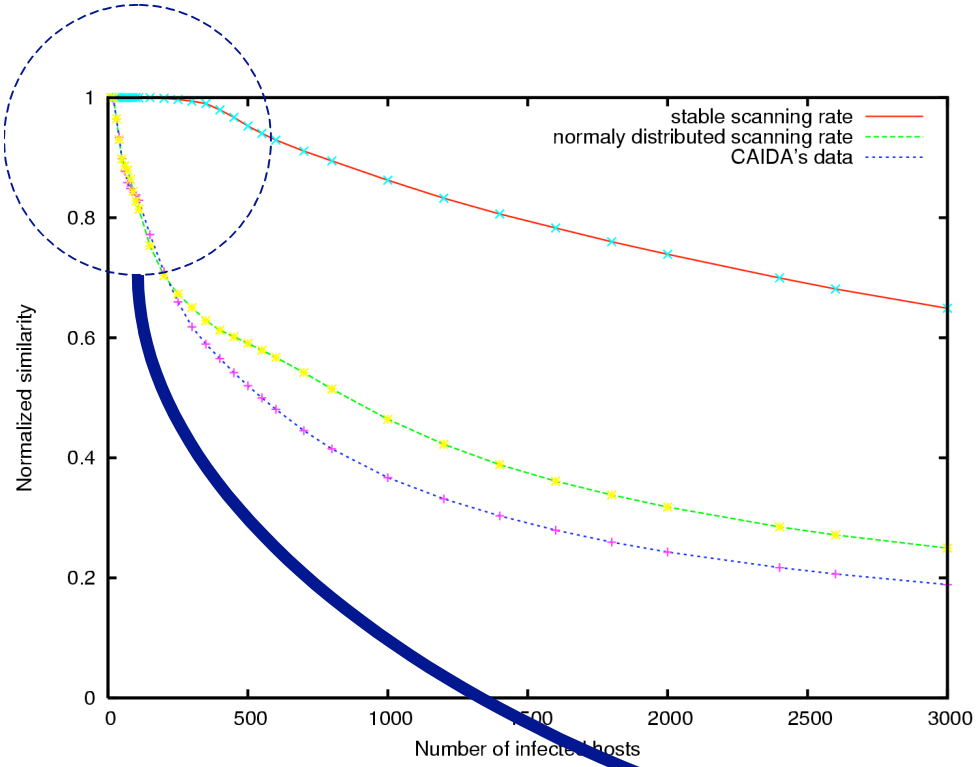
/16 view is completely useless!

Effect of non-homogeneous scanning

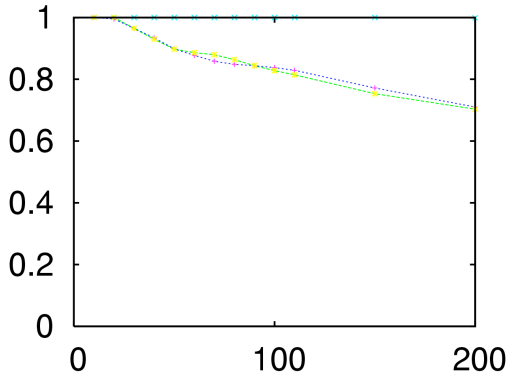


Scanning rate distribution derived from CAIDA's dataset

So, of what good is this?



Who cares what happens after the first 200 infections :-)



Problem Statement and Goals

Consider a uniform scanning worm with scanning rate s and vulnerable population size V and a monitor with effective size M.

- To what extent can a network monitor trace the infection sequence back to patient zero by observing the order of unique source contacts?
- For worms that start with a hitlist, can we use network monitors to detect the existence of the hitlist and determine its size?

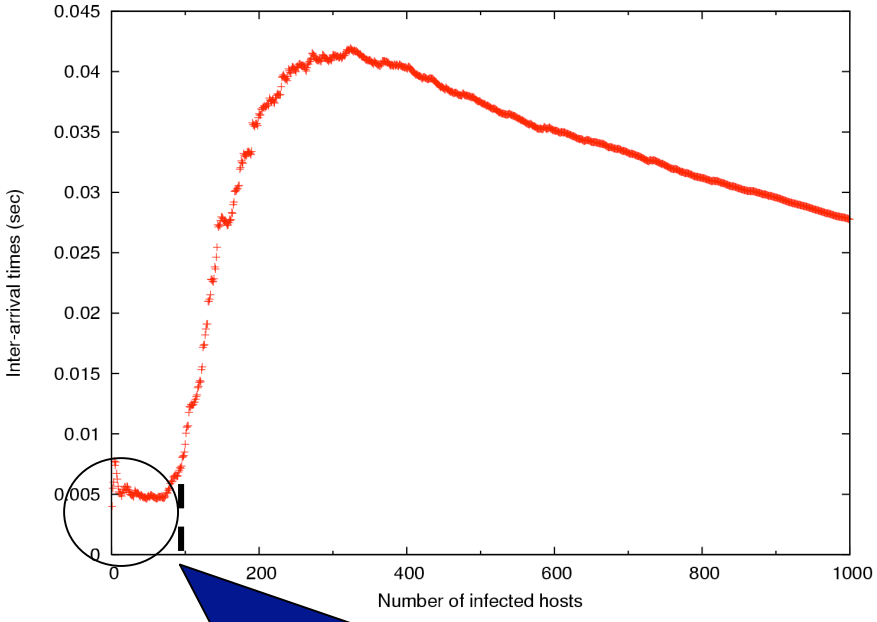
What if the worm starts with a hit-list?

- Hit-lists are used to
  - Boost initial momentum of the worm
  - (Possibly) hide the identity of patient zero

Trick: Exploit the pattern of inter-arrival times of unique sources contacts at the monitor to infer the existence and the size of the hitlist

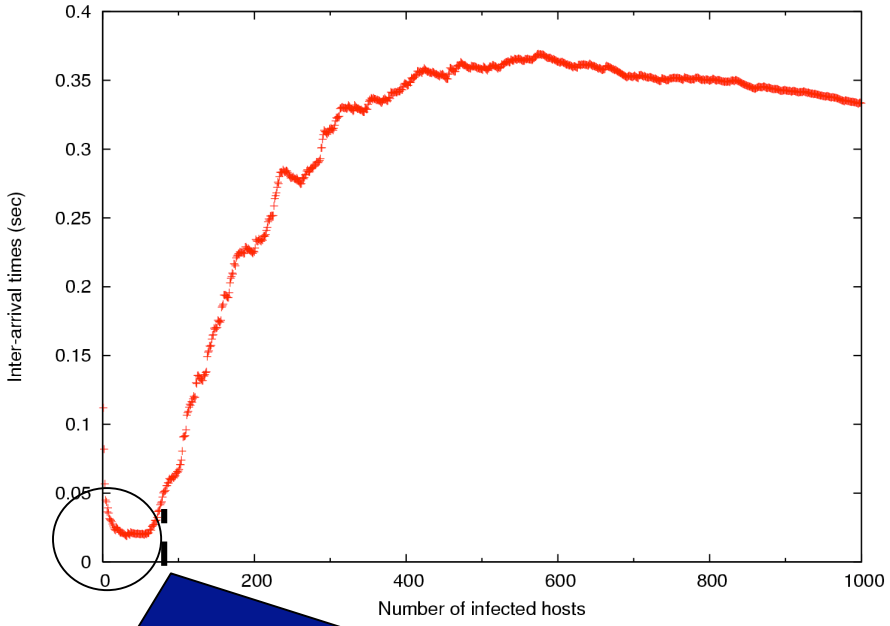
Hit-list detection and size estimation

Simulation (H = 100)



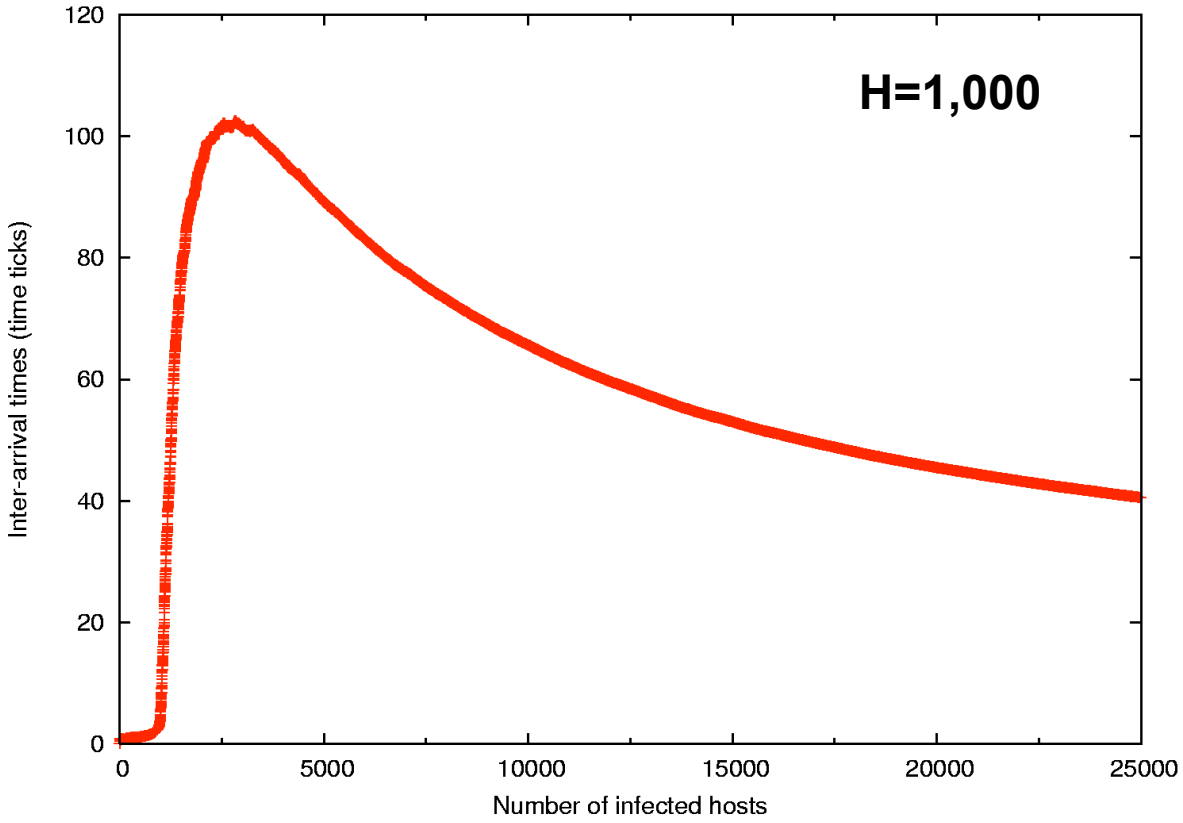
Pattern Change around the hit-list boundaries H = 100

Witty Worm (CAIDA)



Estimated hit-list
H aprox. 80
80% in the same /16
88% belong to the same institution

Will we always see this pattern?



- Same pattern was noticed also when varying population size and with non-homogeneous scanning rates.

Why is that?

- With a hit-list of size h_0 the average worm infection time T_{in} should be less than T_d / h_0

$$\log\left(1-\frac{1}{(V-h_0)}\right) \leq \frac{\log(1-\alpha)\log\left(1-\frac{1}{2^{32}}\right)}{\log\left(1-\frac{M}{2^{32}}\right)}$$

- With a /8 monitor there is no h_0 that can satisfy this inequality
  - Of course, for uniform scanning worms