Rapid Detection of Botnets through Collaborative Networks of Peers

Final Oral Examination

David J. Malan
School of Engineering and Applied Sciences
Harvard University

Michael D. Smith, Advisor

15 May 2007

Background

- **Worm:** Malicious software that propagates on its own (rapidly).
- **Bot:** A worm that can be controlled remotely.
- **Zombie:** A computer that's infected with a bot.
- **Botnet:** A group of zombies infected with the same bot.

Why Botnets Exist

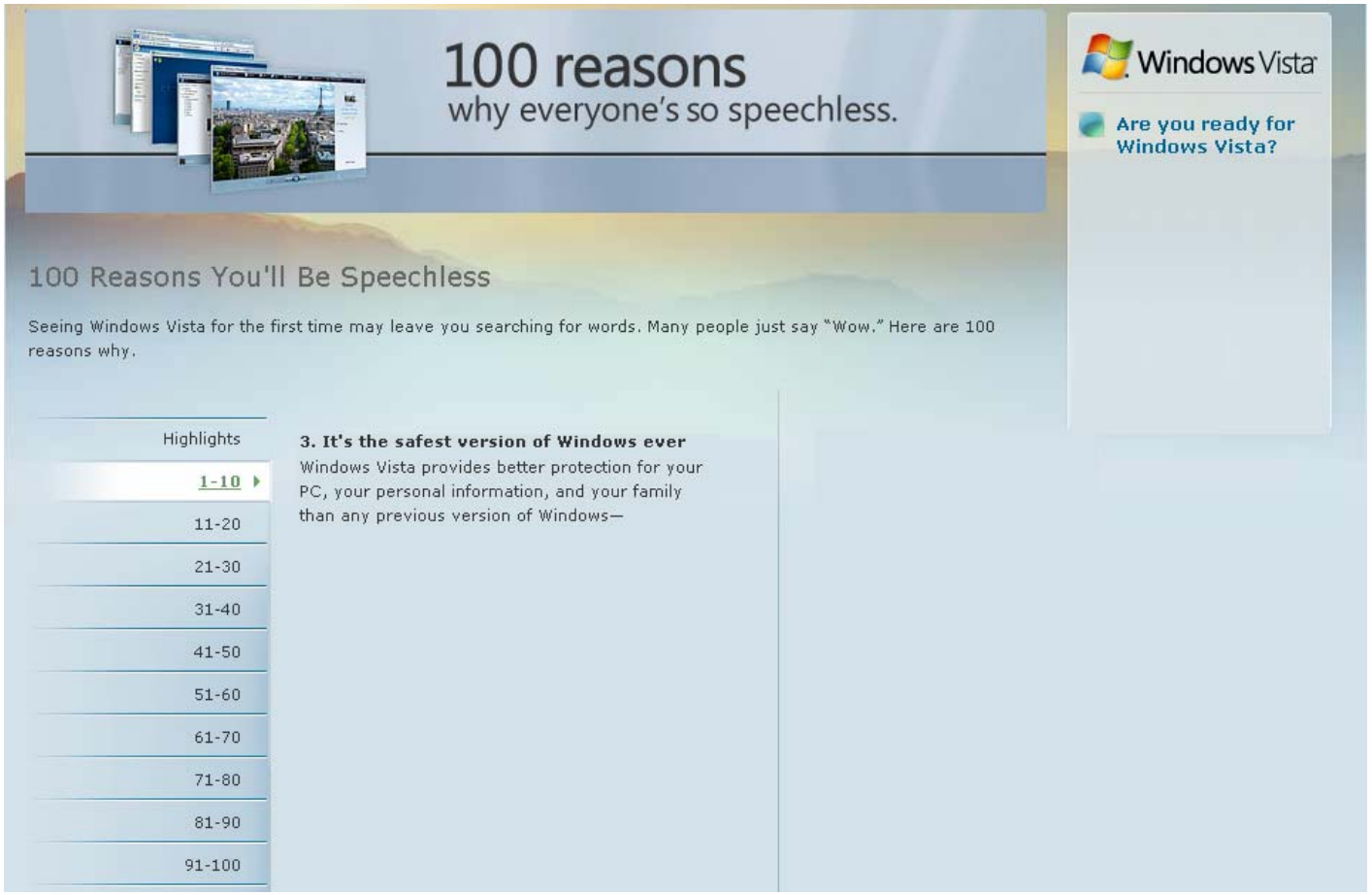
Botnets exist because we're not very good at keeping our systems secure.

Why Systems Are Insecure

Systems run software.

- Bugs
- Complexity
- Monocultures
- Interconnectivity

Aren't We Secure?



Adapted from http://www.microsoft.com/windows/products/windowsvista/100reasons.mspx.

We Aren't.

And adversaries need to find just one of these holes.

Microsoft Windows Cursor And Icon ANI Format Handling Remote Buffer Overflow Vulnerability
19 April 2007

Microsoft Windows Graphics Rendering Engine EMF File Privilege Escalation Vulnerability
19 April 2007

Microsoft Windows Graphics Rendering Engine GDI Local Privilege Escalation Vulnerability
19 April 2007

Microsoft Windows CSRSS HardError Messages Denial of Service Vulnerability
12 April 2007

Microsoft Windows CSRSS MSGBox Remote Code Execution Vulnerability
12 April 2007

Microsoft Windows CSRSS CSRFinalizeContext Local Privilege Escalation Vulnerability
12 April 2007

Microsoft Windows Vista Teredo UDP Nonce Spoofing Weakness
4 April 2007

Microsoft Windows Vista Neighbor Discovery Spoofing Vulnerability
4 April 2007

Microsoft Vista Spoofed LLTD HELLO Packet Security Restriction Bypass Vulnerability
4 April 2007

Microsoft Windows Vista LLTD Mapper EMIT Packet Remote Denial Of Service Vulnerability
4 April 2007

Microsoft Windows Vista Teredo Protocol Insecure Connection Weakness
4 April 2007

Microsoft Windows Vista LLTD Responder Discovery Packet Spoofing Vulnerability
4 April 2007

Microsoft Vista Spoof On Bridge HELLO Packet Security Restriction Bypass Vulnerability
3 April 2007

Microsoft Windows Vista ARP table Entries Denial of Service Vulnerability
2 April 2007

Adapted from http://www.securityfocus.com/bid.

Insecurities Mean Trouble

They allow adversaries to install bots on our systems.

Botnets Are Profitable

Zombies represent cycles and bandwidth.

- Distributed Denial-of-Service (DDoS) Attacks
- Keyloggers
- Relay of Spam
- Click Fraud
- …

Botnets Are Profitable

Profits grow with botnet's size. Profits grow over time.

- Number of Zombies
- Lifetime of Botnet (*i.e.*, Time before Detection)

Detection Isn't Easy

Zombies' behavior resembles "normal" activity.

- Spam is just email
- Clicks are still clicks

Detection of Botnets

Reduces to detection of bots across peers.

I focus on worms.

But worms are too fast for human response.

Detection of Worms

We can build a system with these characteristics.

- Automated
- Rapid
- High in true positives
- Low in false positives

Detection of Worms

We can live with these realities.

- Bugs
- Complexity
- Monocultures
- Interconnectivity

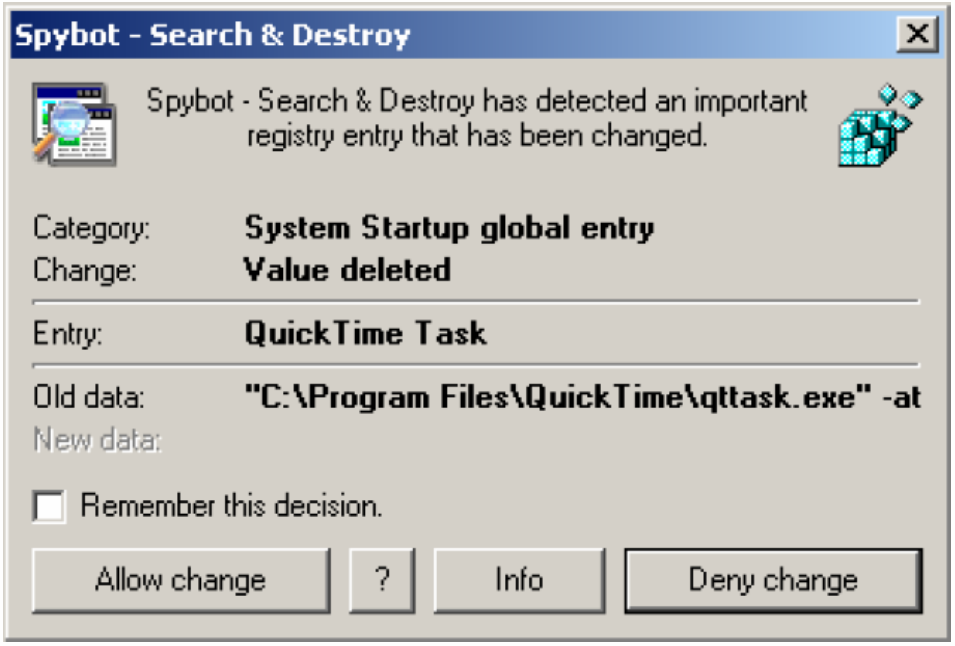
How Not To Detect Worms

Today's detectors are commonly...

- Signature-Based
  - Fast; false positives unlikely
  - But **takes time** to craft signatures;
    can be defeated by metamorphic/polymorphic worms
- Behavior-Based
  - Less susceptible to defeat by transformations
  - Faced with some anomalous action, tend to block that action (which risks **false positives**) or await user's judgement (which interferes with usability)

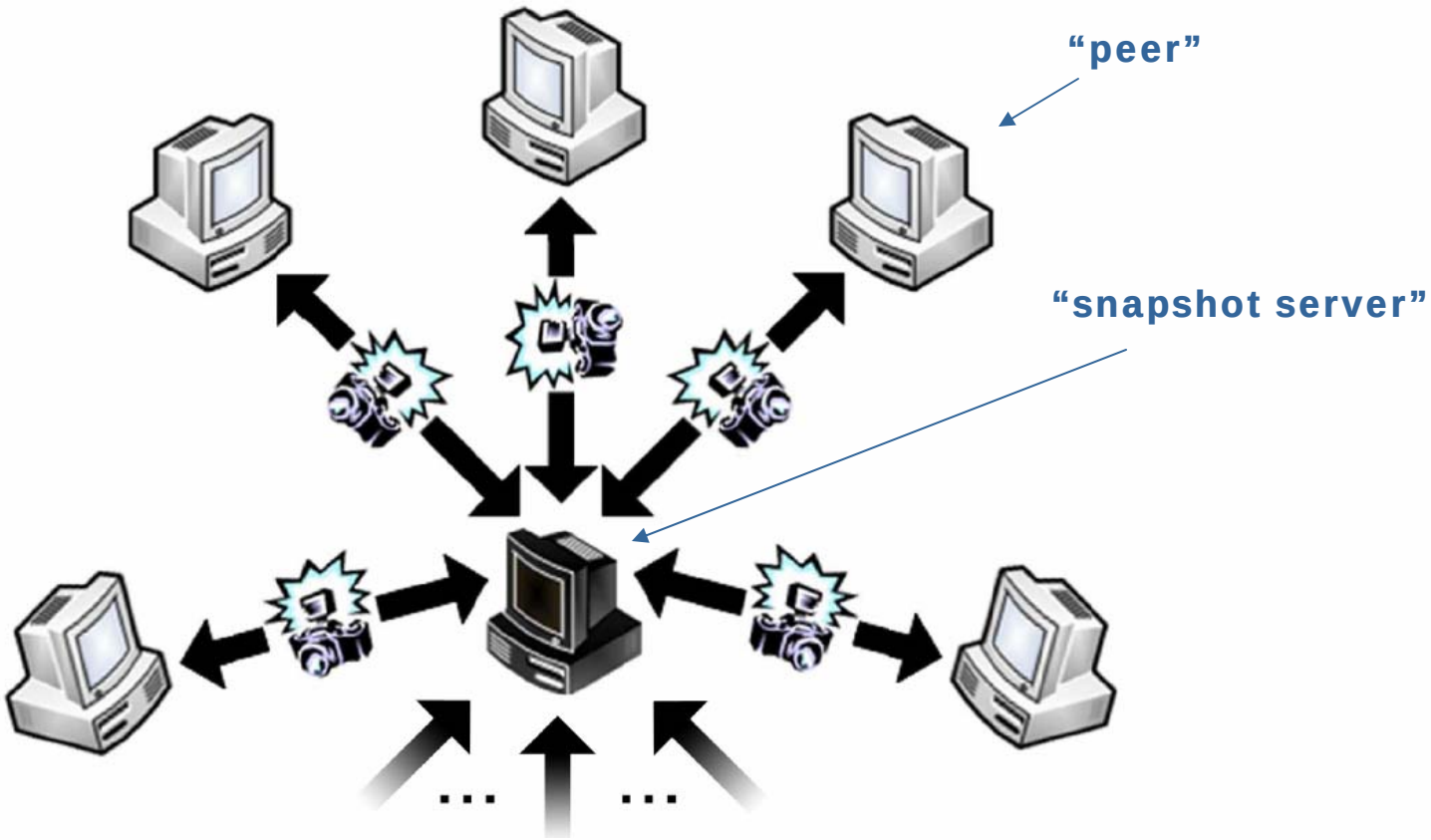
False Positives

We want to avoid them.



How To Detect Worms

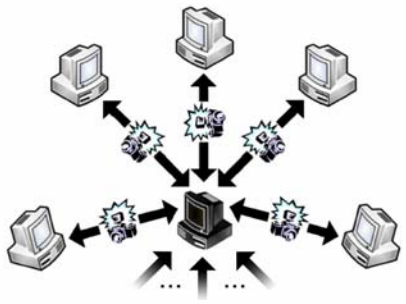
Host-Based, Collaborative Detection



How To Detect Worms

My intrusion-detection system (IDS) is peer-based.

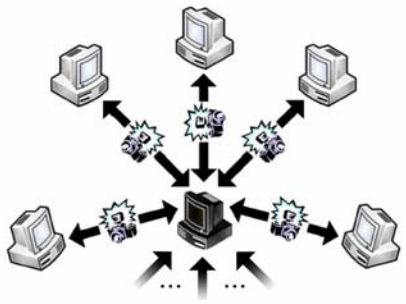
- Time isn't something we have
  - I spend space instead of time—space in the form of peers
- False positives aren't something we want
  - I redefine **anomalous behavior** as correlation among otherwise independent peers' behavior



Hypotheses

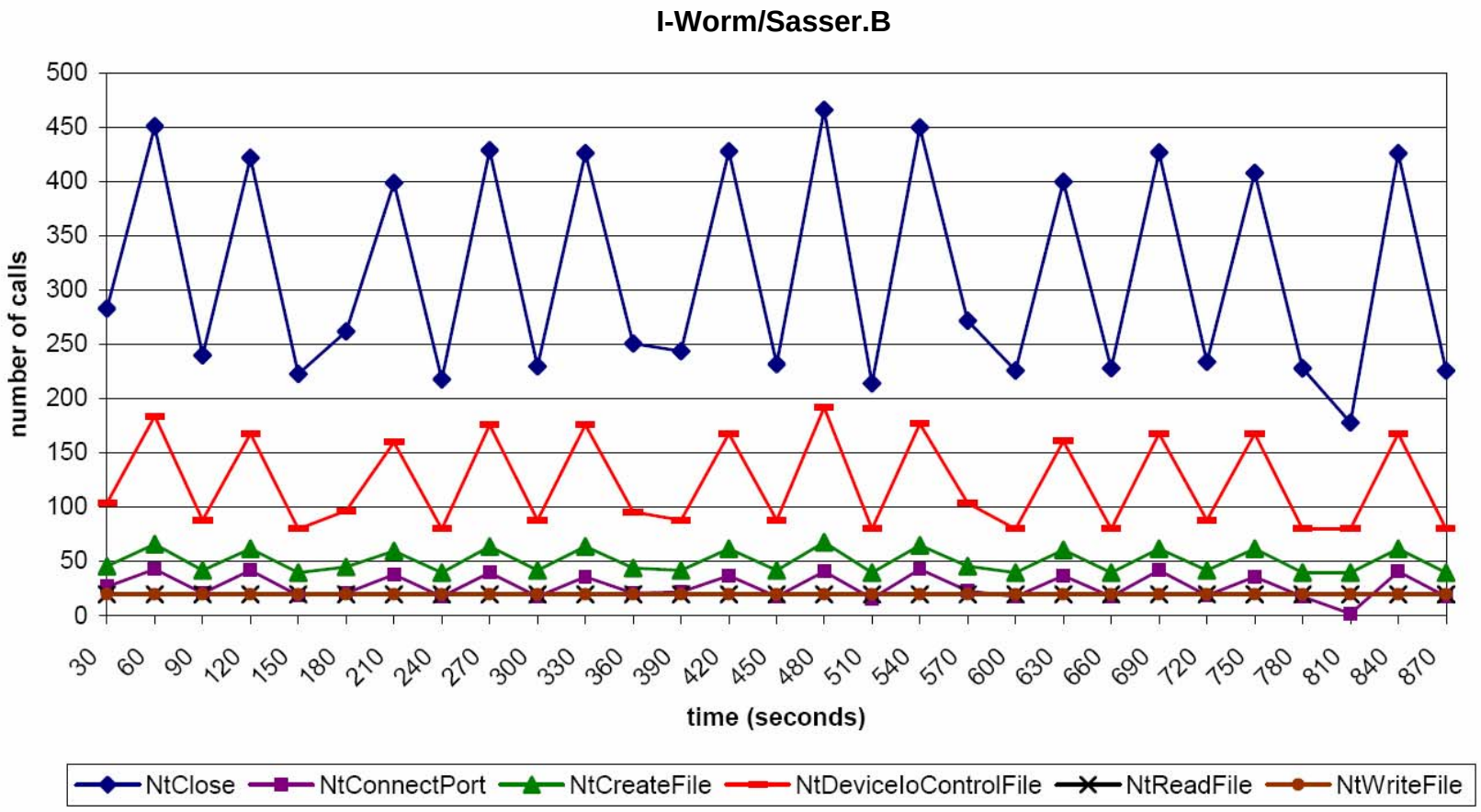
Motivation for Collaborative Detection

- Worms are small, simple, and periodic.
- At any given moment, we are unlikely to see independent hosts behaving almost exactly the same unless triggered by some external threat.
  - distributed applications
  - popular applications



Worms Are "Temporally Consistent"

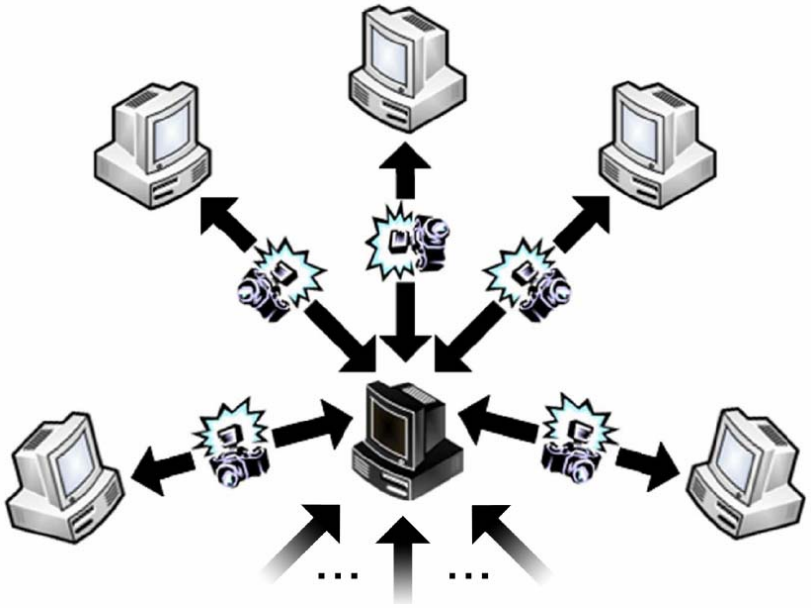
Their behavior is similar over time.



Collaborative Detection

Can it work?

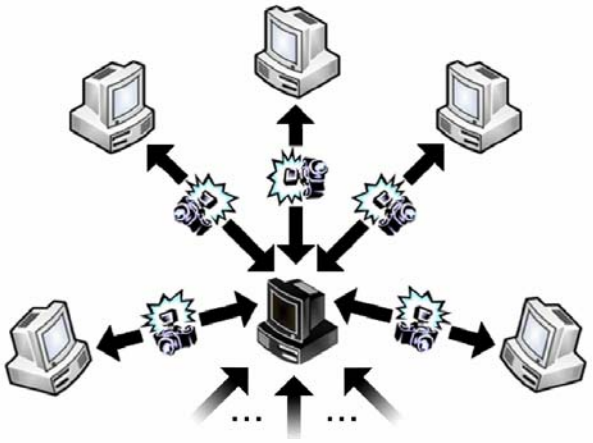
- Automated
- Rapid
- High in true positives
- Low in false positives



Contributions

Collaborative detection can work.

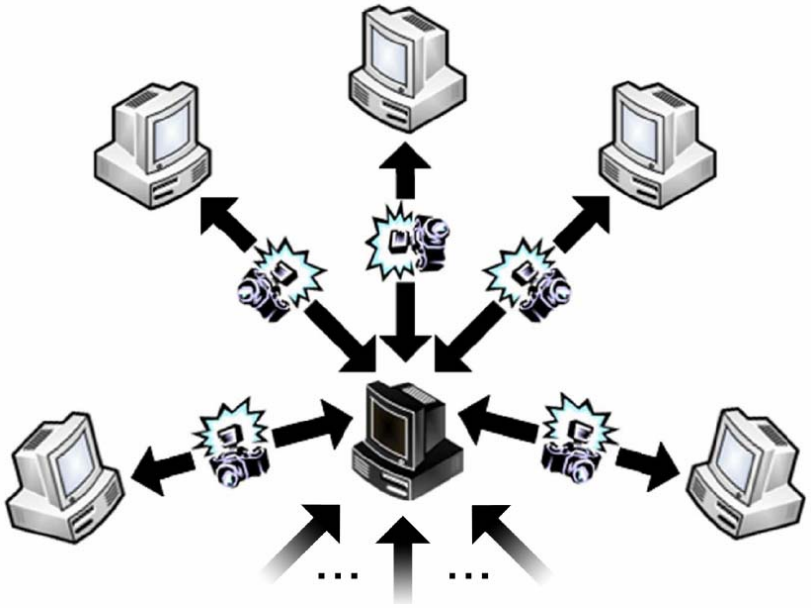
1) I present an architecture that leverages collaboration among peers to detect worms. It is automated, rapid, high in true positives, and low in false positives. And it scales.
2) I provide empirical validation that we can indeed detect behaviors, and thus bots, across peers.
3) I demonstrate that we can, in this domain, tolerate bugs, complexity, monocultures, and interconnectivity.



How to Validate

Let's focus on true positives first.

- Automated
- Rapid
- High in true positives
- Low in false positives



How to Validate

Let's focus on true positives first.

- Automated
- Rapid
- High in true positives
- Low in false positives

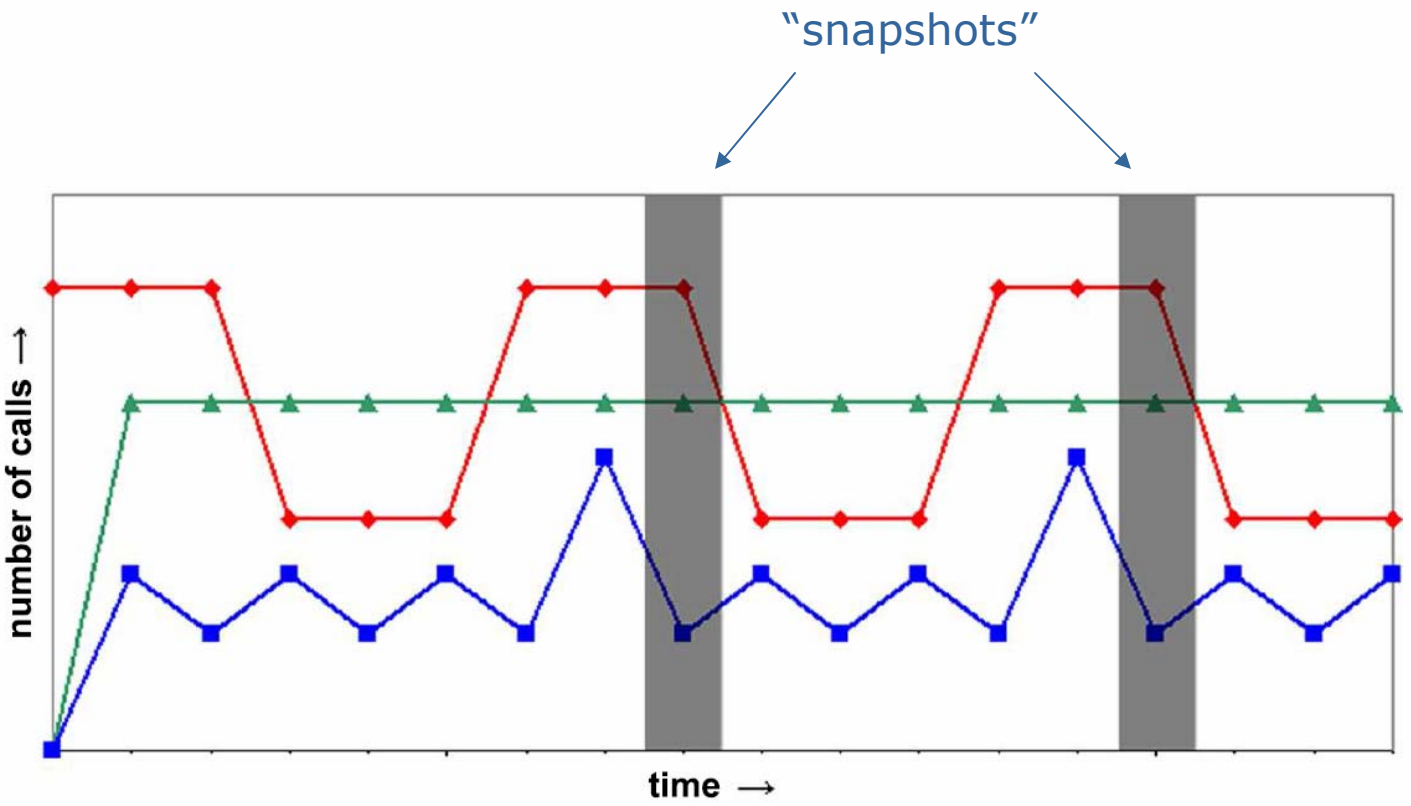
How to Validate

What I Need First

- Model for Behavior
- Metric for Comparing Behavior

Model for Behavior

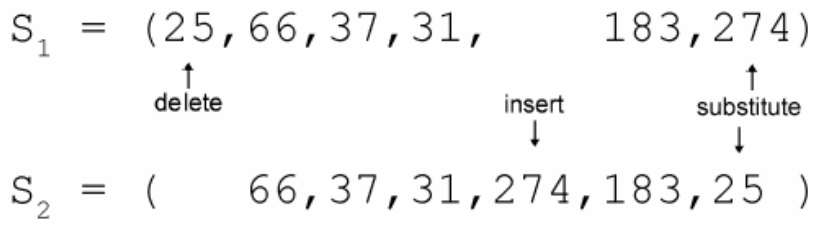
Snapshots are sets of system calls (ordered or unordered).



How to Measure Similarity

Edit Distance v. Intersection

- If snapshots are **ordered**, we can measure similarity in terms of **edit distance**.



= 3/6 = **50% similar**

- If snapshots are **unordered**, we can measure similarity in terms of **intersection**.

S_1 = (25,31,37,66,183,274)

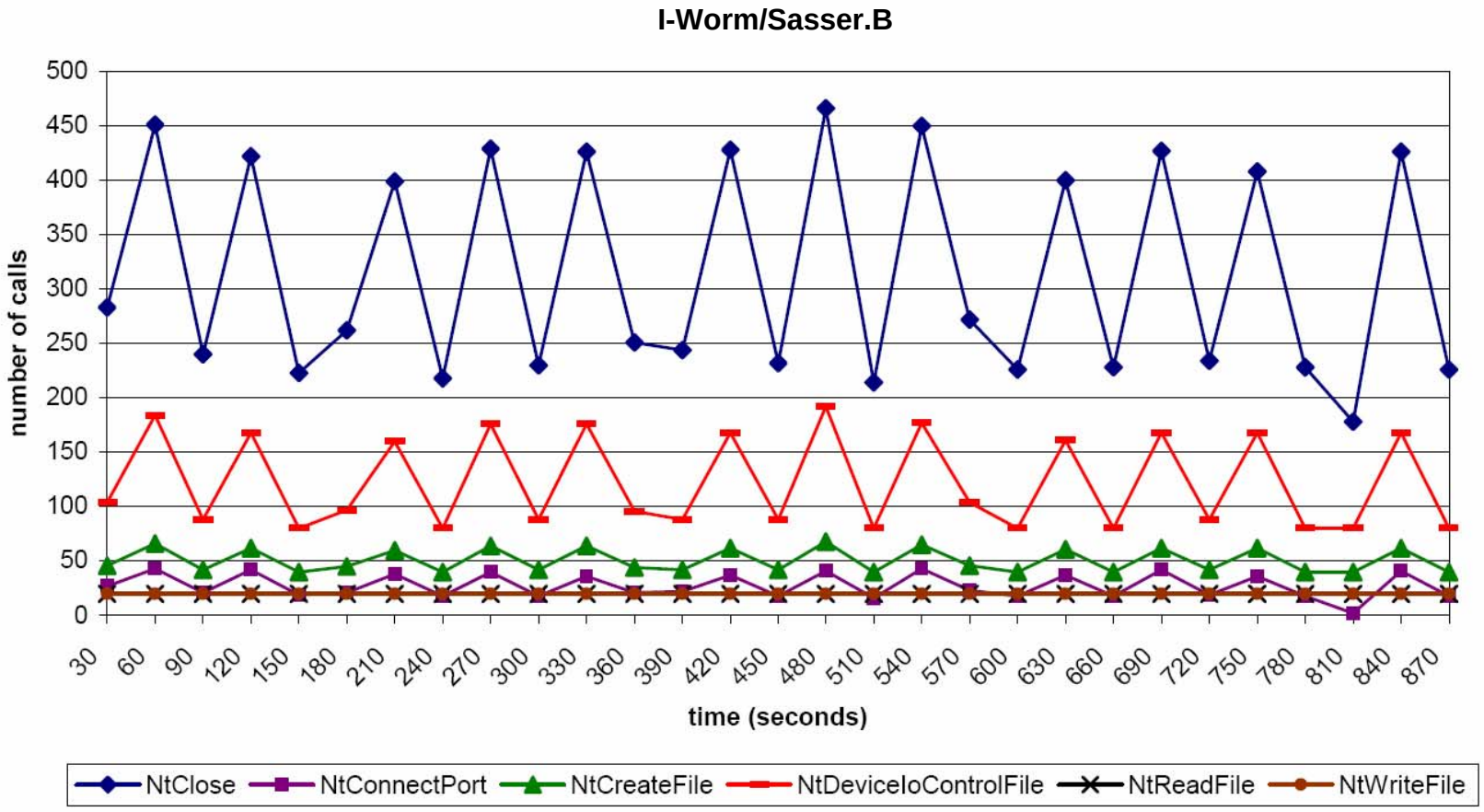
S_2 = (25,31,37,66,183,274)

$S_1 \cap S_2$ = (25,31,37,66,183,274)

= 0/6 = **100% similar**

Measuring Temporal Consistency

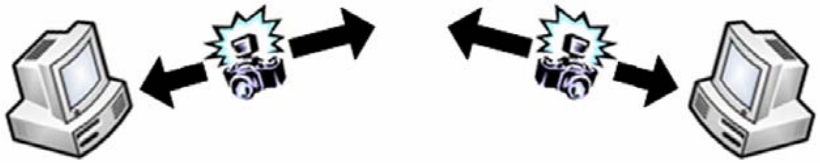
τ = % of snapshots that are similar over time



Exploiting Temporal Consistency

Can we use τ to detect worms and only worms?

1) Are worms **temporally consistent**?
2) Are non-worms **temporally consistent**?



Methodology

I used traces of 9 worms and 25 non-worms to simulate 2 peers.

Worms	Non-Worms	
	Commercial Applications	Windows XP SP2 Binaries
I-Worm/Bagle.Q	Adobe Photoshop 7.0.1	alg.exe
I-Worm/Bagle.S	Microsoft Access XP SP2	csrss.exe
I-Worm/Jobaka.A	Microsoft Excel XP SP2	defrag.exe
I-Worm/Mydoom.D	Microsoft Outlook XP SP2	dfrgntfs.exe
I-Worm/Mydoom.F	Microsoft Powerpoint XP SP2	explorer.exe
I-Worm/Sasser.B	Microsoft Word XP SP2	helpsvc.exe
I-Worm/Sasser.D	Network Benchmark Client 1.0.3	lsass.exe
Worm/Lovesan.A	Nullsoft Winamp 5.094	msmsgs.exe
Worm/Lovesan.H	Windows Media Encoder 9.0	services.exe
	WinZip 8.1	spoolsv.exe
		svchost.exe
		wmiprvse.exe
		winlogon.exe
		wscntfy.exe
		wuauclt.exe

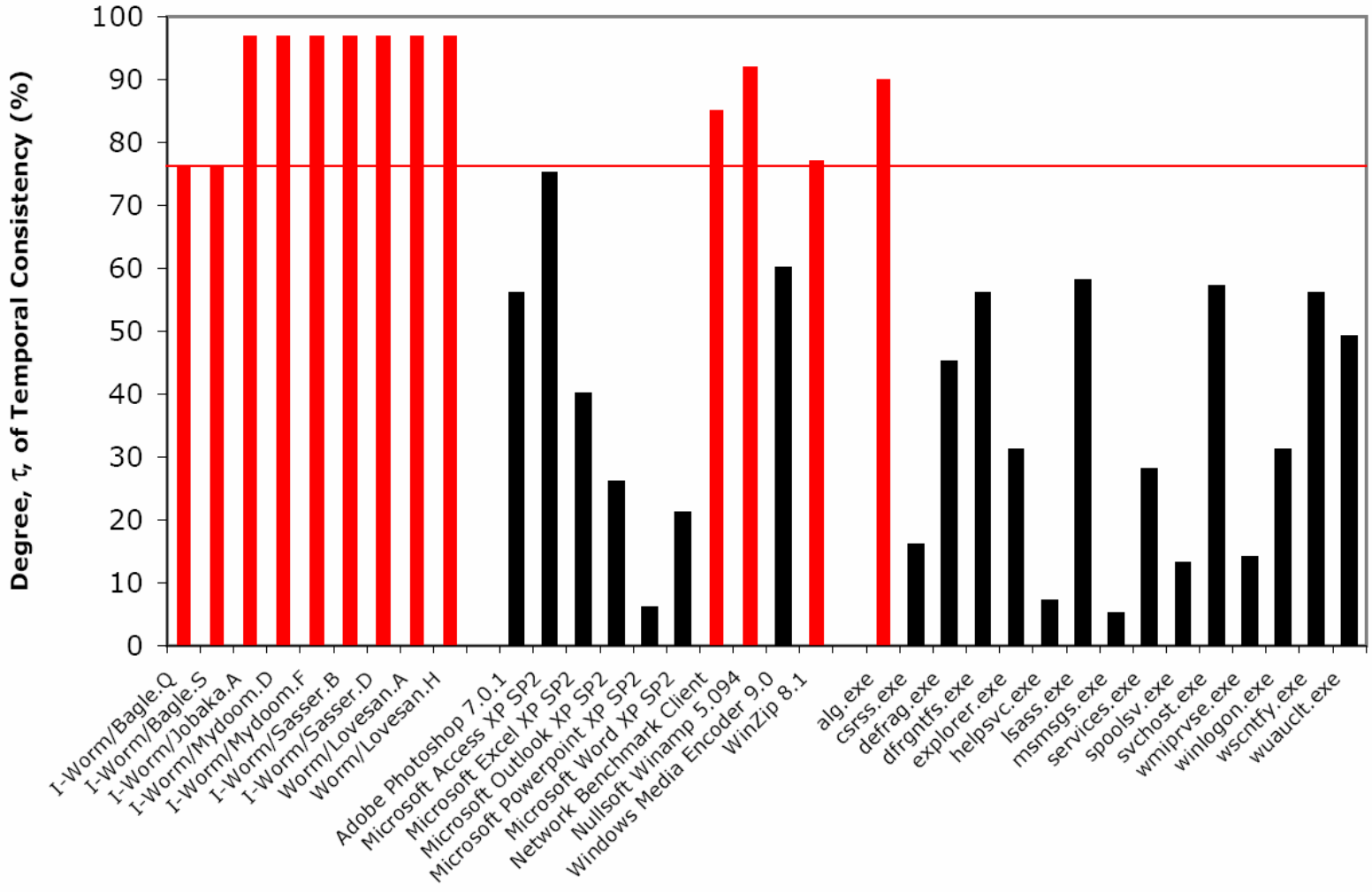
Temporal Consistency of Worms

Using intersection, worms' τ are indeed high.

	edit	isect
I-Worm/Bagle.Q	11%	76%
I-Worm/Bagle.S	11%	76%
I-Worm/Jobaka.A	50%	97%
I-Worm/Mydoom.D	81%	97%
I-Worm/Mydoom.F	31%	97%
I-Worm/Sasser.B	54%	97%
I-Worm/Sasser.D	97%	97%
Worm/Lovesan.A	98%	97%
Worm/Lovesan.H	95%	97%

Temporal Consistency of Non-Worms

Some non-worms have high (*i.e.*, worm-like) τ.



Results

High τ implies high true positives

1) Worms are very temporally consistent.
   - Two peers can decide that they are, more likely than not, executing a worm between 76% and 97% of the time.
2) But some non-worms are too.
   - Network Benchmark Client, Winamp, WinZip, and `alg.exe` boast $\tau > 76\%$.
     - We need to filter these potential false positives.
     - Could whitelist, but let's consider generalized techniques.

False Positives

Why are they worrisome?

1) If we mistake a popular non-worm for a worm,
   we might declare an outbreak when there is none.
2) If we confuse a non-worm on one host with a worm on another,
   we might overstate an outbreak's severity.

Avoiding False Positives

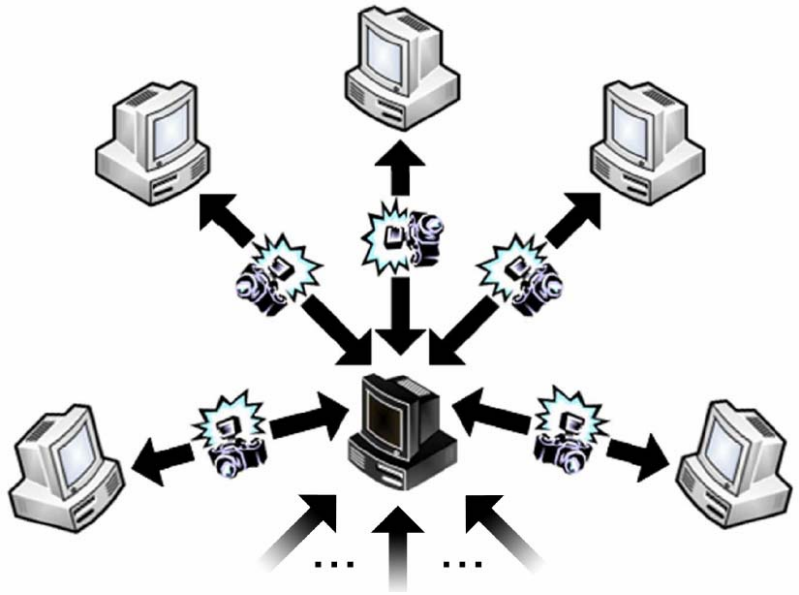
Worms have properties that (most) non-worms don't.

- Relying only on τ puts us at risk of false positives.
  - Recall Network Benchmark Client, Winamp, WinZip, and `alg.exe`.
- Filtering by *r* (syscalls/second) eliminates some false positives.
  - *E.g.*, `alg.exe` only executes 1 syscall per second!
- Filtering by *r'* (sockets/second) eliminates even more.
  - Now only Network Benchmark Client appears worm-like.

Studying False Positives

I further assessed the problem in the wild.

- Deployed prototype on **29 real-world hosts** running Windows XP with Service Pack 2
- Deployed prototype to one snapshot server.
- Monitored and analyzed **10,776 processes**, including **511 unique non-worms** (873 unique versions)



Non-Worms in the Wild

Simple filters help distinguish non-worms from worms.

τ = degree of temporal consistency

r = syscalls/sec

r' = sockets/sec

- All of my worms boasted $\tau \geq 76\%$, $r \geq 64$, and $r' \geq \delta$.
- Less than 1% (3 of 511) of my non-worms exceed these thresholds and remain potential false positives.
  - `ApntEx.exe`
  - `explorer.exe`
  - `OUTLOOK.EXE`

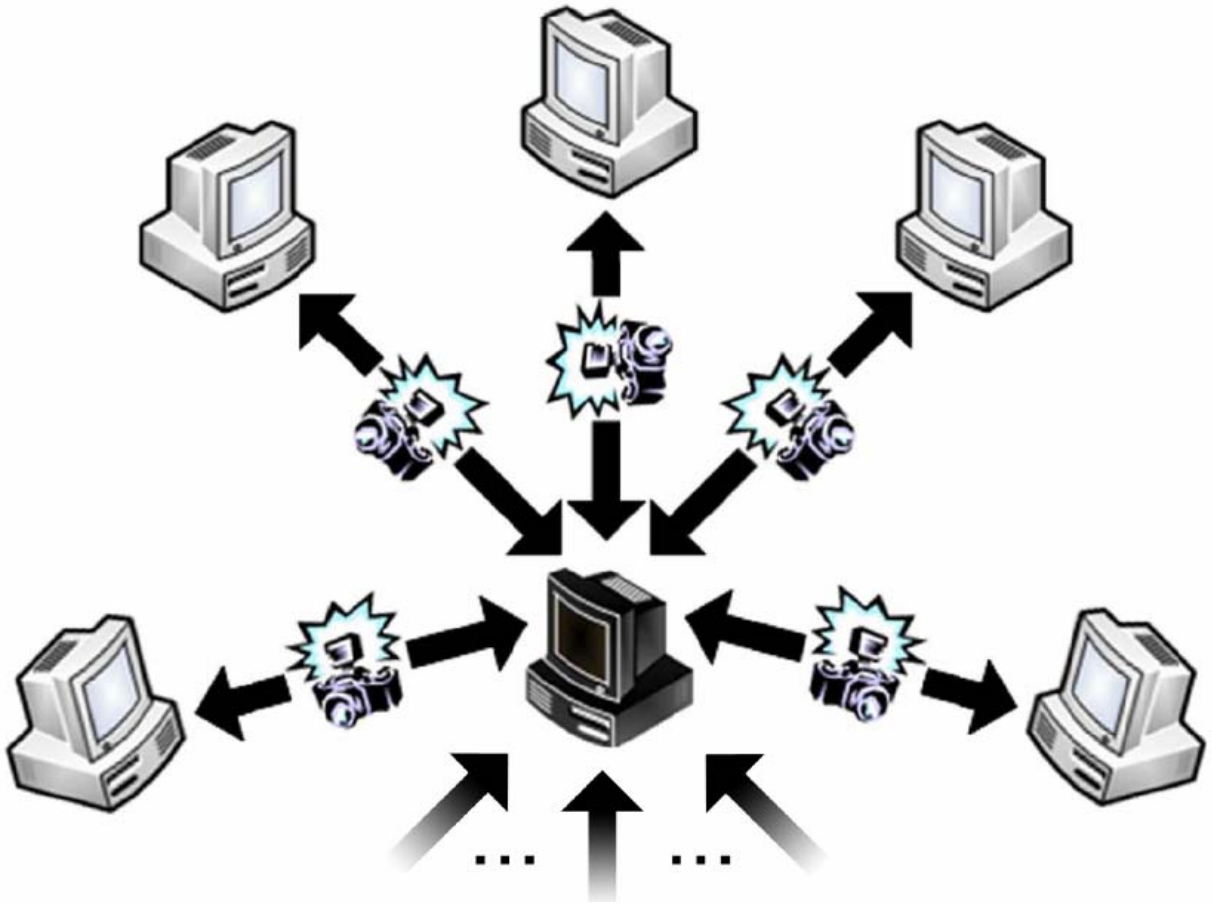
Detecting Worms across Peers

Can we detect behaviors across peers?

- We don't know (or need to know) τ a priori. I post-processed my data to measure it for 9 worms and 25 + 511 non-worms, but τ isn't actually measured by architecture itself.
- But high τ should lead to high rates of recognition: if a process is executing on *n* peers and we detect it on *m*, **rate of recognition is *m*/*n***.

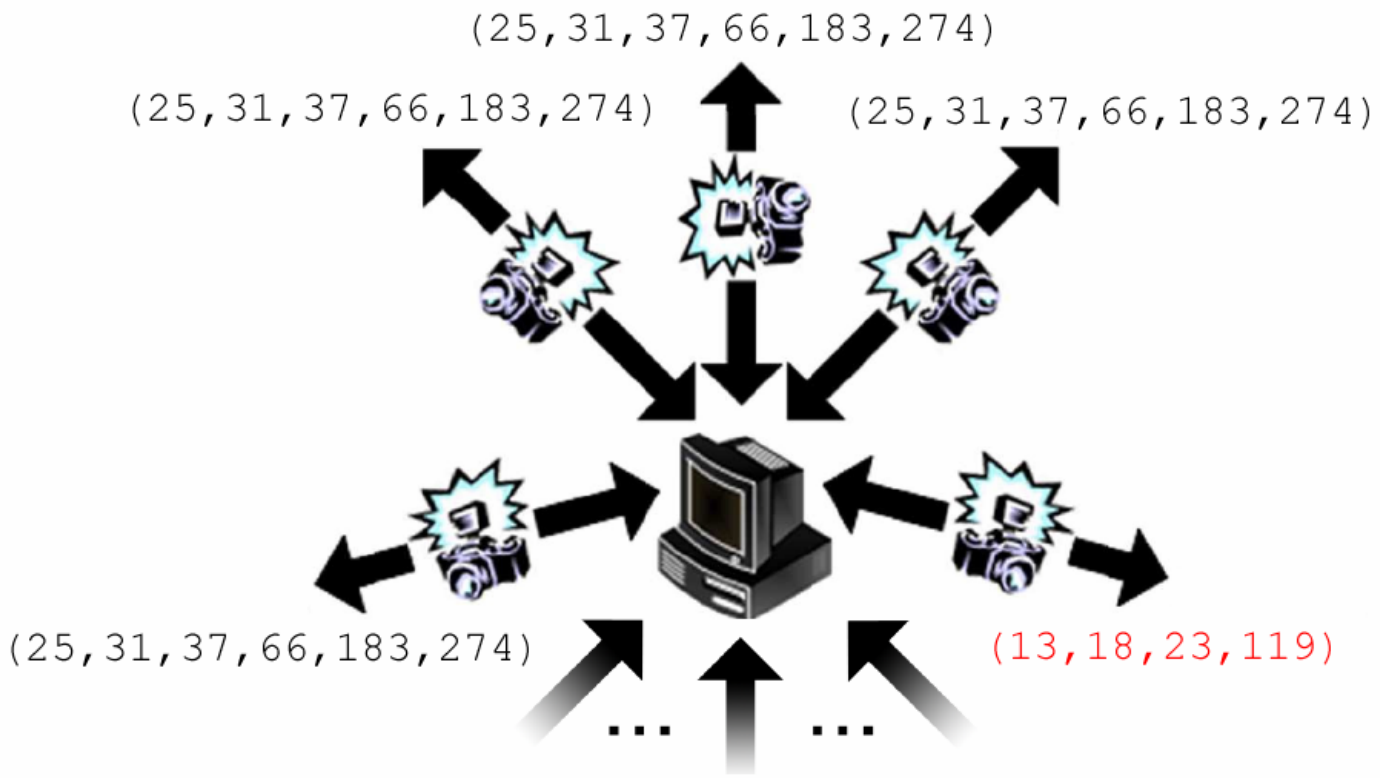
Detecting Worms across Peers

Can we detect behaviors across peers?



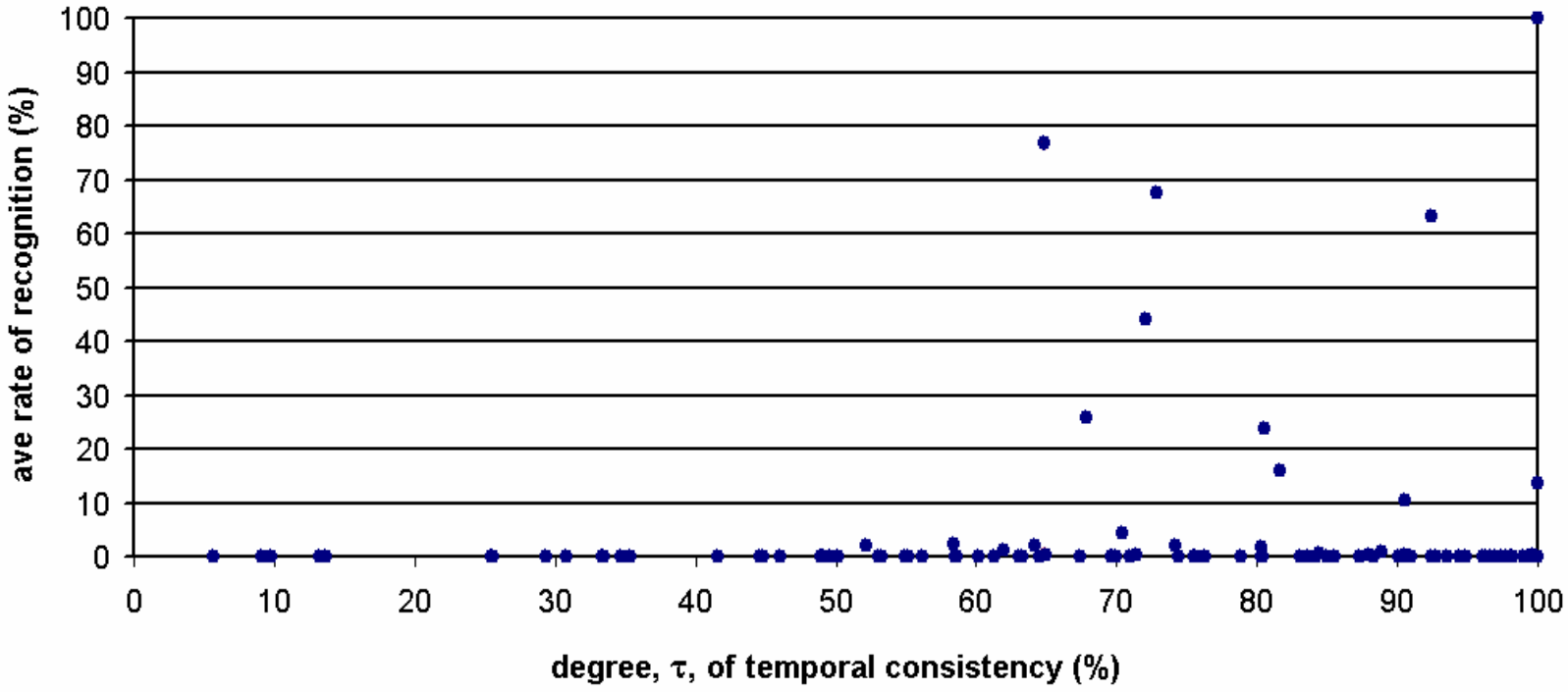
Detecting Worms across Peers

***m*/*n* = 4/5, but *m*/*n* isn't necessarily an infection rate!**



Detecting Worms across Peers

Higher τ implies higher rates of recognition (m/n).



Related Work

Collaborative detection just isn't used (yet).

- McAfee, Symantec, *et al*.
  - Traditional signature- and behavior-based defenses.
- Somayaji *et al*. (pH)
  - Monitoring processes for unexpected patterns of system calls.
- Karp *et al*. (Autograph, Polygraph)
  - Automated generation of network-traffic signatures.
- …

Conclusions

Collaborative detection can work.

- Collaborative networks of peers allow us to detect behaviors across systems.
  - By generalizing anomalous behavior as similar behavior across peers, detection is **automated**.
  - Behavior can be summarized and compared efficiently, so detection is **rapid**.
  - We can exploit worms' temporal consistency to achieve **high rates of true positives**.
  - Even simple filters produce **low rates of false positives**.
- Collaborative networks of peers tolerate bugs, complexity, monocultures, and interconnectivity.

Rapid Detection of Botnets through Collaborative Networks of Peers

Final Oral Examination

David J. Malan
School of Engineering and Applied Sciences
Harvard University

Michael D. Smith, Advisor

15 May 2007