Statistical Methods for Network and Computer Security

David J. Marchette

marchettedj@nswc.navy.mil

Naval Surface Warfare Center

Code B10

< > - +

A Few Areas for Statistics

- Estimating the number of denial of service attacks on the Internet.
- Determining the operating system of the machine that sent a packet (passive operating system fingerprinting).
- Profiling users and detecting masqueraders.
- Some areas I won't talk about (but am intersested in):
  - Profiling applications (by packet statistics).
  - Modeling virus/worm propagation.
  - Other methods for intrusion detection.
  - Graph-theoretical methods of analyzing attacker behavior.

Denial of Service Attacks

- A class of denial of service attacks allow detection via passive monitoring.
- These attacks result in the victim sending out packets (responses) to random hosts on the Internet.
- By monitoring these unsolicited packets, we can estimate the number of attacks.
- This allows us to learn about attacks as they are happening *w*ithout the need for sensors at the victim and with no reliance on victim reporting.

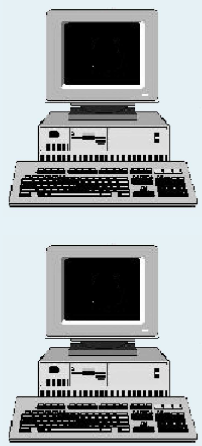
< > - +

Backscatter Cartoon

Victim

Typical Denial of Service Attack: Syn Flood.

Attacker floods the victim with connection requests.

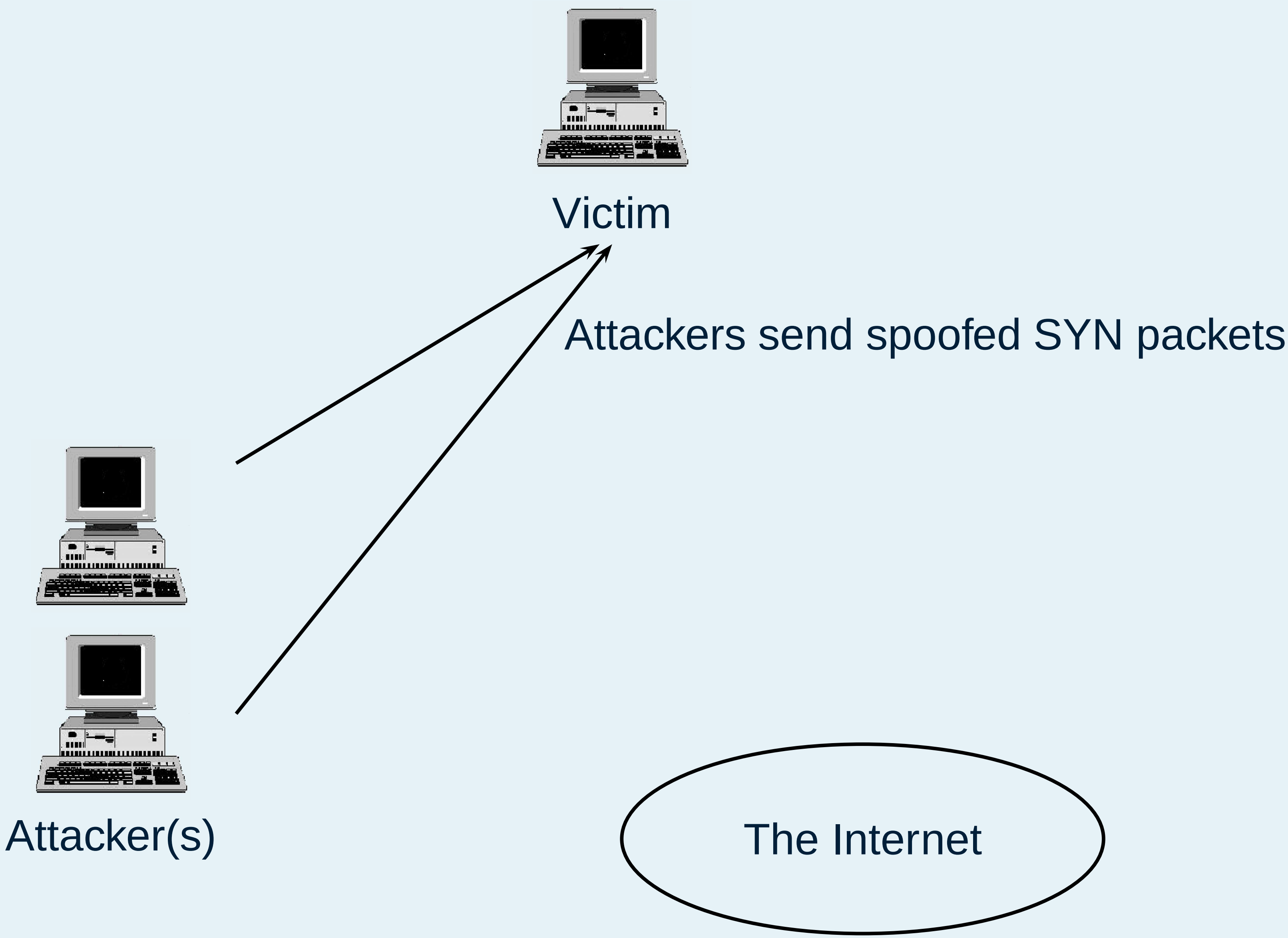


Attacker(s)

The Internet

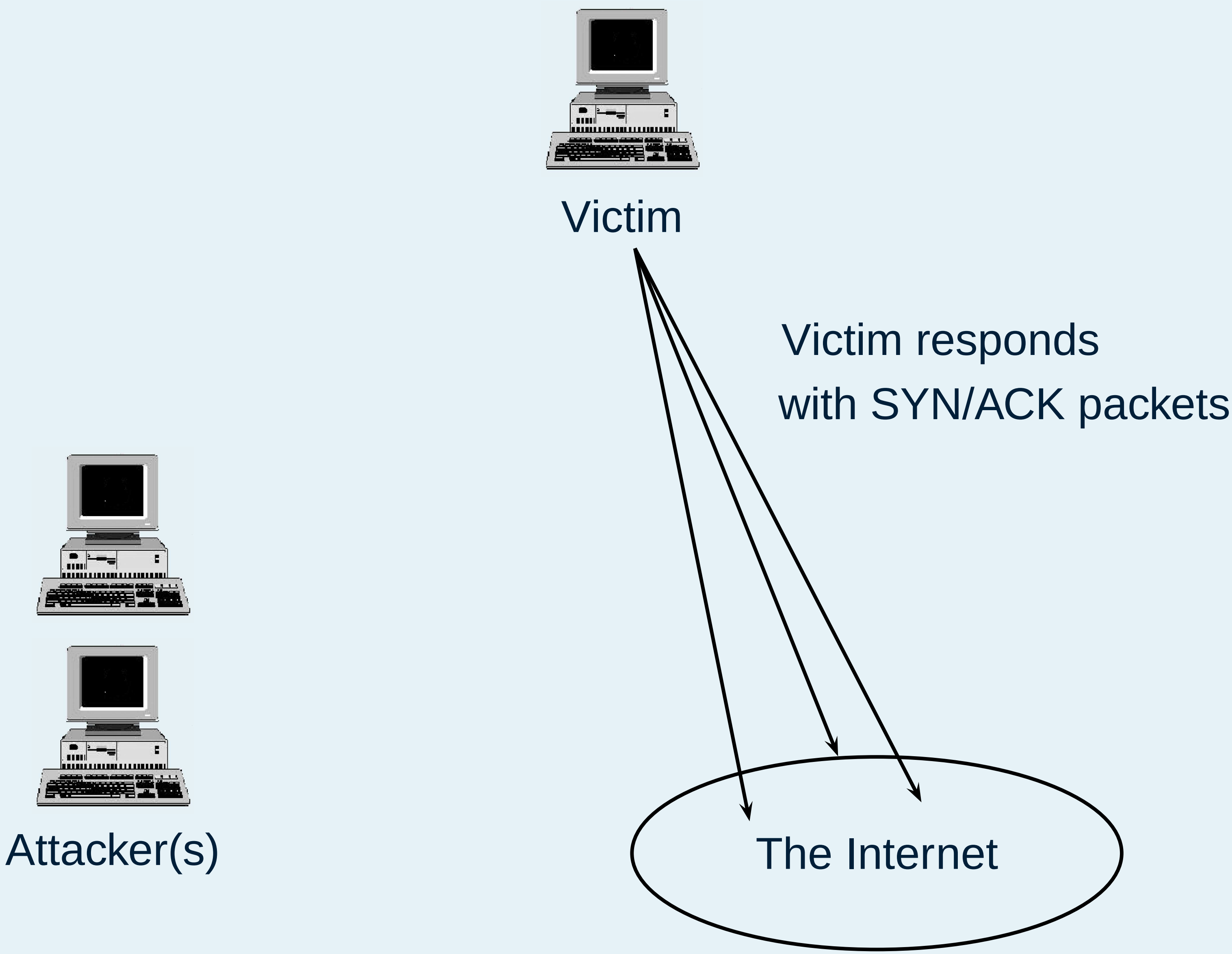
< > - +

Backscatter Cartoon

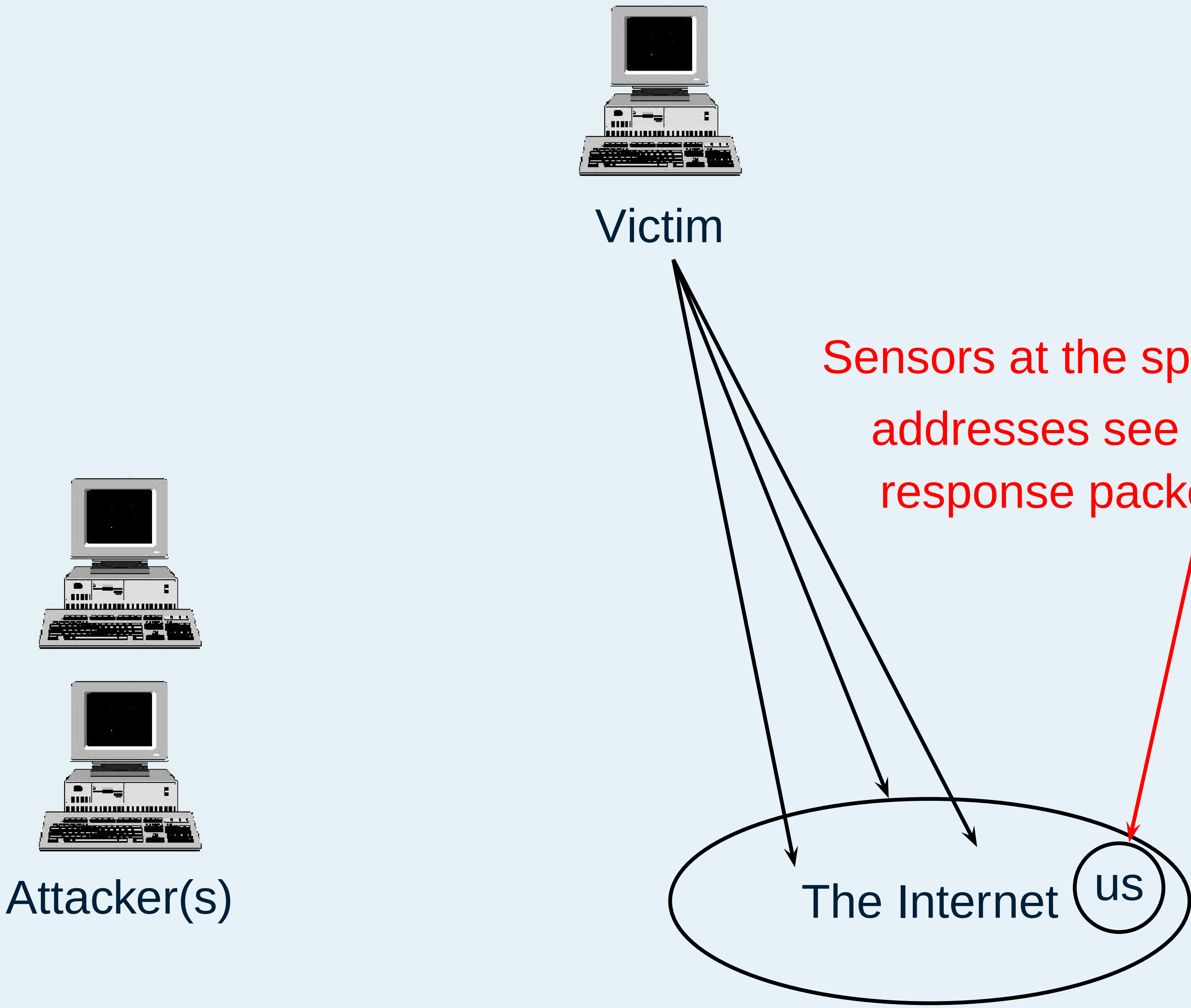


< > - +

Backscatter Cartoon



Backscatter Cartoon



Some Observations

- We observe a subset of the response packets.
- Estimation requires that we understand the model.
- Perusal of attack software indicates that random (uniform) selection of (spoofed) IP addresses is common.
- Unsolicited response packets may also be an attack against the monitored network.
- We'd also like to estimate the effect of the attack.
- The attacks evolve, and this approach may need modification or be invalid in the future.

Assumptions

- We assume (and perusal of some attack code bears this out) that the spoofed IP addresses are selected at random (independently, identically distributed, uniformly from all possible addresses).
- Given this, we can estimate the size of the attack, the number of attacks we are likely to miss, etc.
- Assume m packets are sent in the attack.
- Assume we monitor n of the $N = 2^{32}$ possible IP addresses.
- Assume no packet loss.

< > - +

Probability of Detecting an Attack

- Then the probability of detecting an attack is:

$$P[\text{detect attack}] = 1 - \left(1 - \frac{n}{N}\right)^m .$$

- The expected number of backscatter packets we detect is $\frac{nm}{N}$.
- The probability of seeing exactly j packets is:

$$P[j \text{ packets}] = \binom{m}{j} \left(\frac{n}{N}\right)^j \left(1 - \frac{n}{N}\right)^{m-j} .$$

- This allows us to estimate the size of the original attack:

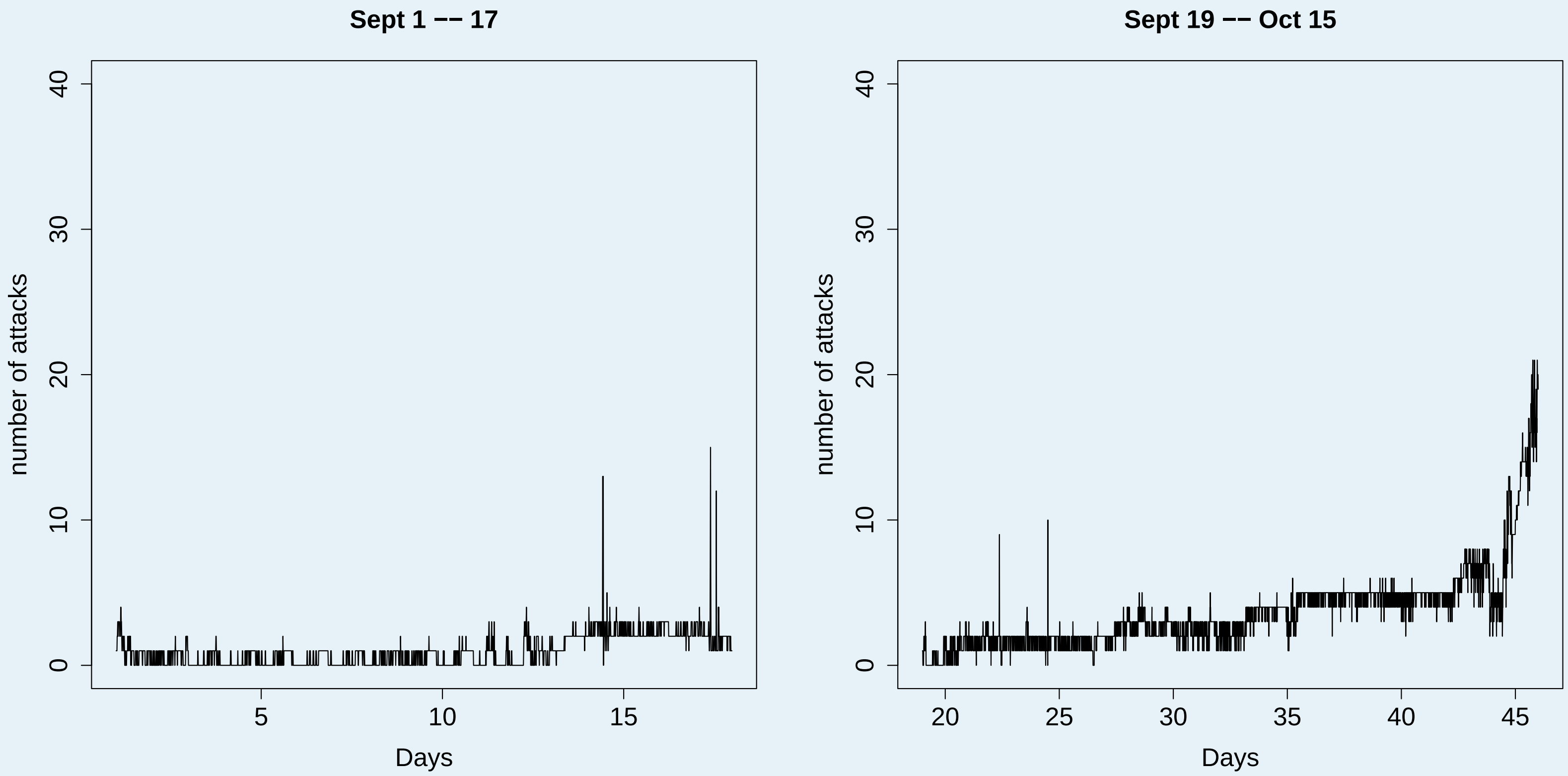
$$\hat{m} = \left\lfloor \frac{jN}{n} \right\rfloor .$$

Modeling and Classification

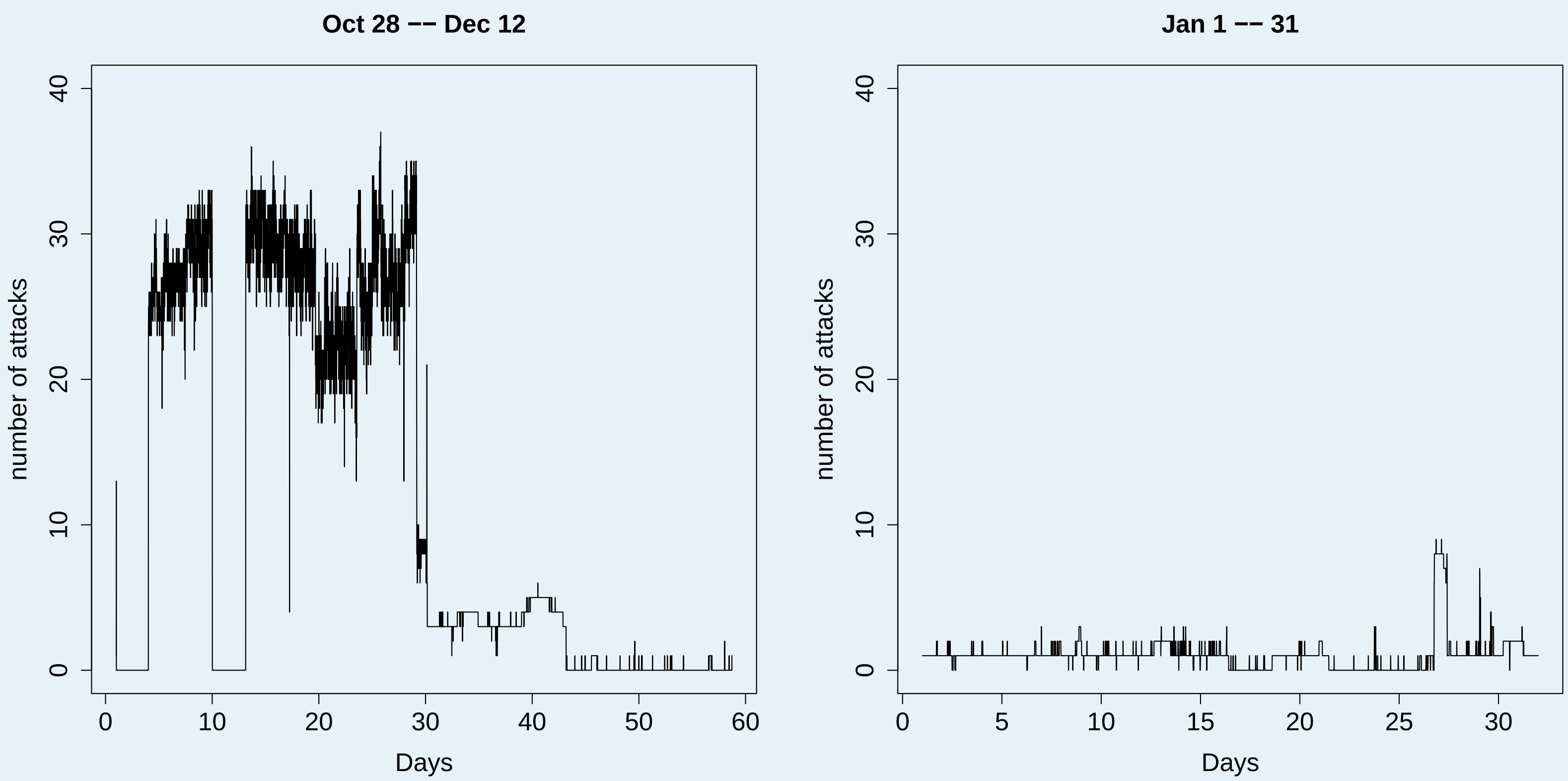
- We need good models of the spoofing process(es).
- These can help classify the attacks (identify the attack code).
- Given these models, we can estimate the size of the attack.
- These models are also necessary to estimate the number of attacks that are not observed at the sensor(s).

< > - +

Number of Attacks Observed



Number of Attacks Observed



< > - +

Comments

- Something happened to change the volume of attacks in mid September.
- These are only “big” attacks (those where the sensor sees more than 10 packets).
- These are only attacks against web servers.
- At the peak, there were more than 30 victims at any one time, over a period of a month.
- By January, things were back to “normal”.
- By doing this type of analysis, we can have a “Internet threat level” monitor that is continuous, essentially instantaneous, and requires no cooperation.

Passive Fingerprinting

- The protocols specify what a host must do in response to a packet or when constructing a packet.
- These specifications are not complete: there are several choices that a computer is free to make.
- These choices are made differently (to some extent) by different operating systems.
- By monitoring these, one can guess the operating system.
- The idea is to examine packets coming to the monitored network, and determine the operating system of the sending machine.

Time To Live

- One such choice is the time to live (TTL) value.
- This is a byte (value 0–255) set when the packet is constructed.
- Each router decrements the TTL.
- If the TTL is 0, the router drops it, and sends an error message.
- Different operating systems choose different default values for TTL.
- We never observe the original TTL: we observe the TTL minus a random number (corresponding to the number of routers in the route it took).

Why Do We Care?

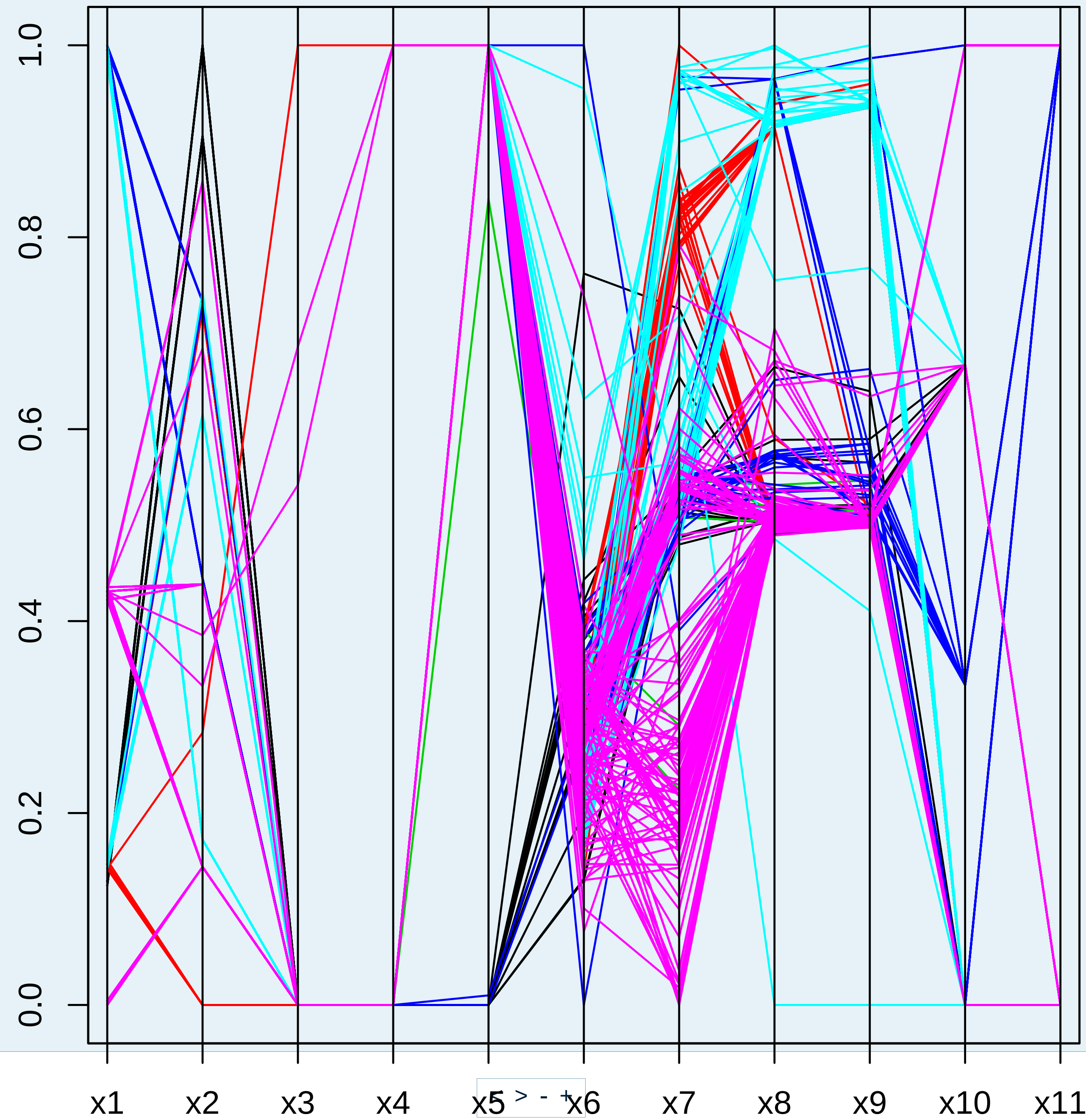
- It's fun.
- Passive validation of the accreditation database.
- A machine that appears to change it's operating system may be evidence of an attack (specially crafted packets).
- The operating system of an attacker can indicate the likely attack software.
- In very rare scenarios this could be used to craft a response.
- This last suggestion probably would only work against a frequent (legitimate) visitor who's OS was known reasonably reliably; it may be illegal.

The Experiment

- Data collected on 3806 machines over a period of about 6 months.
- Features such as: mean TTL, mean type-of-service, window size, IP ID and sequence number increment, min/max source port, number of IP options, which options, whether DF flag set.
- Data split into a training and a test set evenly (split so that each OS had the same number in training and test).
- Operating system classed as: Generic DOS, Irix, Generic Apple, Mac, Solaris, Windows.
- OS designation comes from an accreditation database (unknown amount of inaccuracy).
- Ran k-nearest neighbor classifiers on training data, best was $k = 3$.

< > - +

Some Data



Some Results 3-NN classifier

	dos	irix	linux	apple	mac	solaris	windows
dos	0	0	0	0	2	0	32
irix	0	16	0	0	0	0	1
linux	0	0	25	0	0	0	0
apple	0	0	0	0	3	0	3
mac	0	0	0	0	31	0	0
solaris	0	0	0	0	0	27	0
windows	1	0	6	0	3	0	1753

Bottom line error: 0.027.

Worst case error: 0.074.

< > - +

Reduced Classes 3-NN

dos $\longrightarrow$ windows

apple $\longrightarrow$ mac.

	windows	irix	linux	mac	solarisw
windows	1786	0	6	5	0
irix	1	16	0	0	0
linux	0	0	25	0	0
mac	3	0	0	34	0
solaris	0	0	0	0	27

Bottom line error: 0.008.

Worst case error: 0.056.

Summary

- Very simple classifier works quite well.
- Windows dominates.
- Better data collection is necessary.
- The sub-classes are available (Windows NT vs 98 vs 2000...).
- Active fingerprinting can determine these quite well.
- Passive fingerprinting is undetectable and adds nothing to the load on the network.

< > - +

Network User Profiling

- Tracking users by their network activity can provide an indication of suspicious or dangerous behavior.
- Network activity involves:
  - Applications used (web, ftp, telnet, ssh, etc.).
  - Servers accessed.
  - Amount of data transfered.
  - Temporal information.
- I do not consider (but could) web pages visited, etc.

Web Servers Visited

- Construct an intersection graph according to the web servers visited by each user.
- The vertices of the graph are the users.
- There is an edge between two users if they visit the same web server (during the period under consideration).
- This is computed over the full time period over which the data were collected (3 months).
- We want to be able to group users, determine if users change their behavior, etc. To determine how much "change" is significant, we need a model.

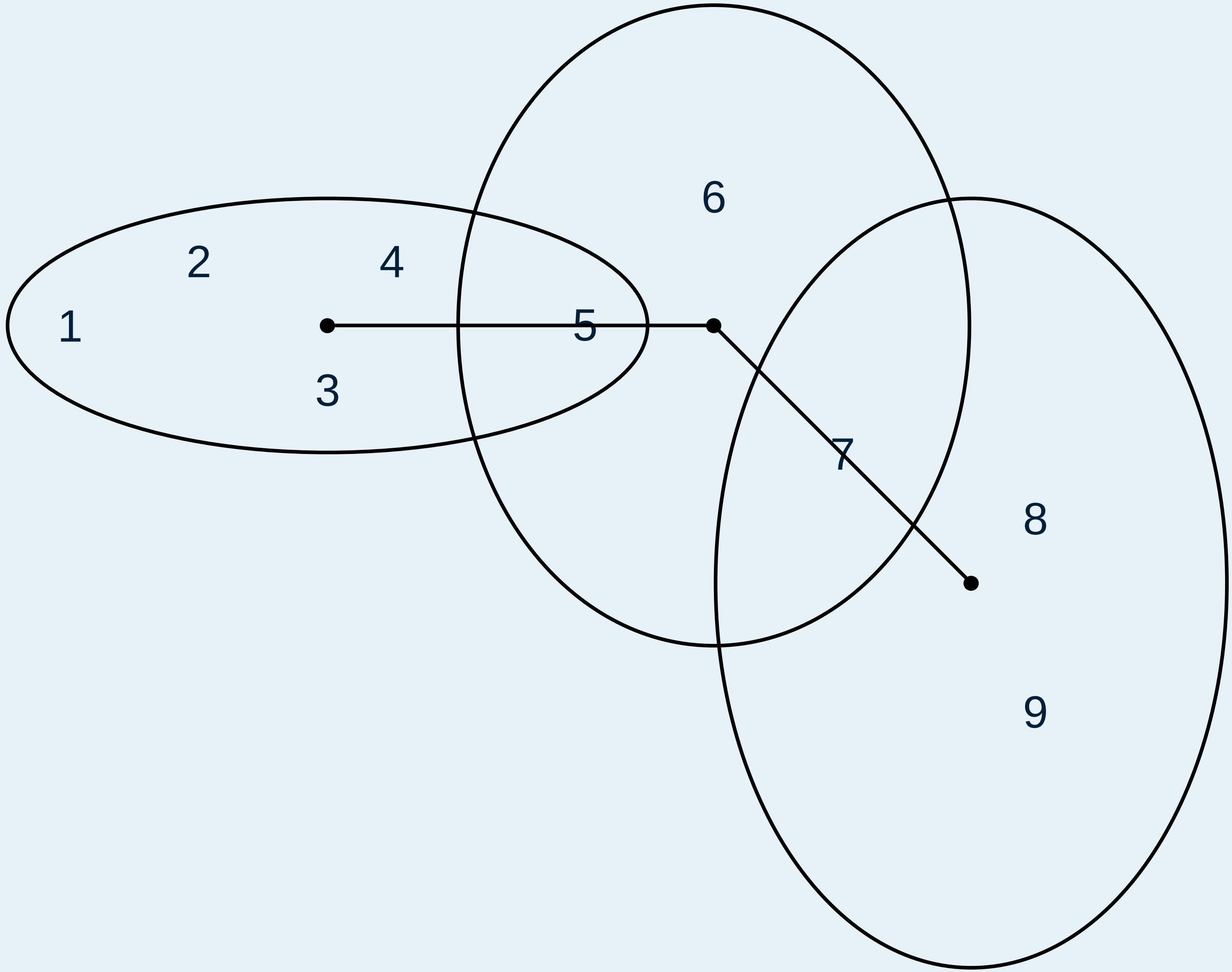
Intersection Graphs

- Given a set S of m elements, the random intersection graph model is as follows.
- Each vertex v_j selects each element of S with probability p, resulting in the random set S_j.
- There is an edge between two vertices $v_i v_j$ if $S_i \cap S_j \neq \emptyset$.

These random graphs will give us a framework for modeling and investigating user behavior.

< > - +

Intersection Graphs



< > - +

Intersection Graphs: Estimation

- Given an intersection graph G with associated sets S_i.
- We want to determine what the original m and p were.
- With this, we have a random graph model and we can test how likely it would be to obtain a graph such as the one we observe, under the random hypothesis.
- Some methods for estimating m and p are suggested by calculations on the random graph.
- We will also borrow some techniques from the literature on estimating animal abundance.

Estimation continued

- One obvious estimate is:

$$\begin{aligned}\hat{m} &= |\cup S_i| \\ \hat{p} &= \frac{\sum |S_i|}{n\hat{m}}.\end{aligned}$$

- This is obviously biased (low in m).
- Note: if $S = \{1, \ldots, m\}$ then we might estimate m as $\max \cup S_i$.

< > - +

Estimation continued

- Let

$$\begin{aligned} s &= \text{size}(G) \\ d &= \text{density}(G) \end{aligned}$$

- It can be shown (Karonski et al. 1999) that

$$E(s) = \binom{n}{2}(1-(1-p^2)^m).$$

- Set $E(s) = s$ and solve, and we have the optimization problem

$$\hat{m} = \underset{m}{\operatorname{argmin}} \left(\sum |S_i|/(nm) - \sqrt{1-(1-d)^{1/m}} \right)^2$$

< > - +

Estimation continued

- Given an intersection graph G with associated sets S_i.
- Mark-Recapture model. Set:

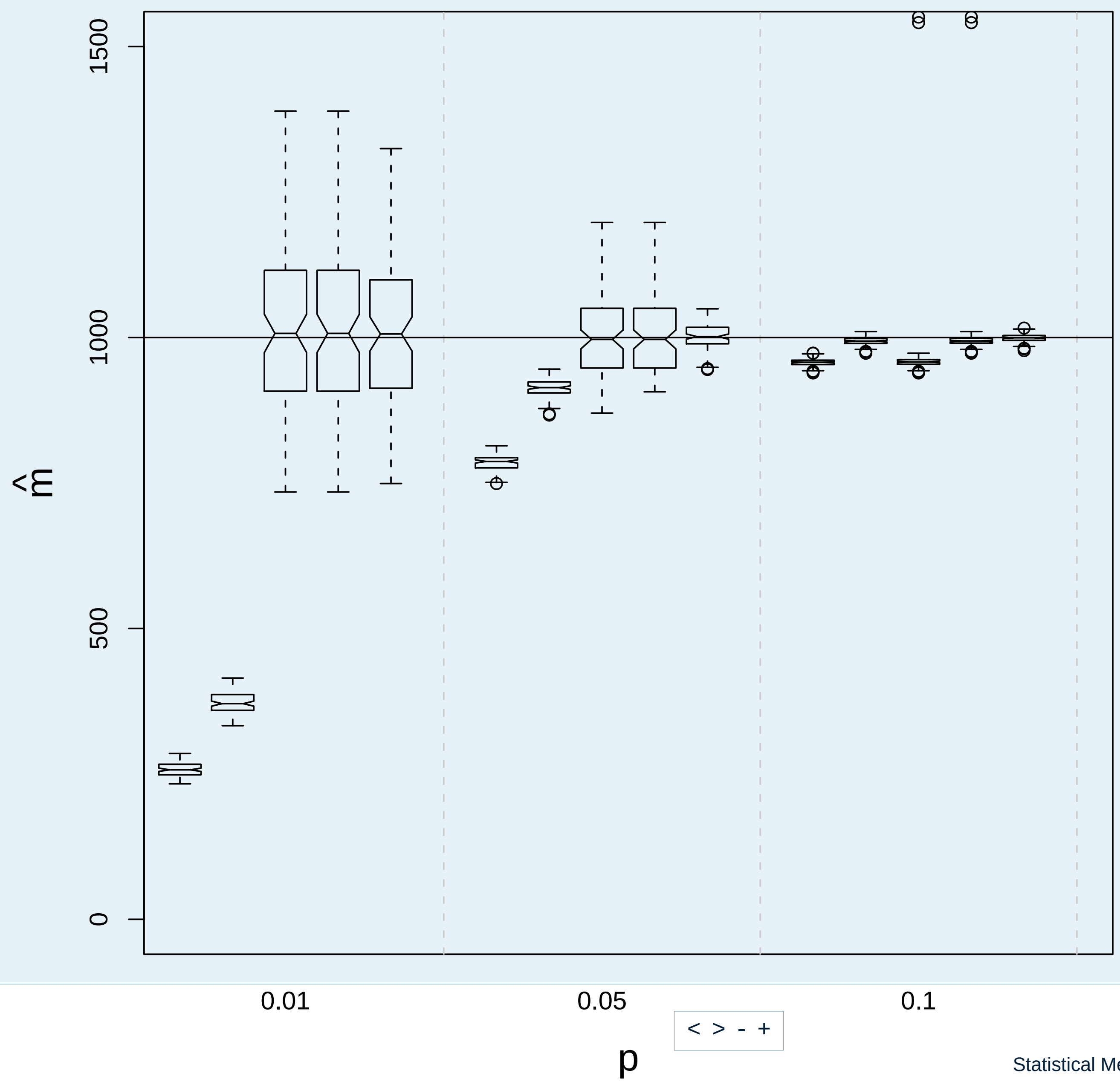
$$\begin{aligned} k_i &= |S_i| \text{ the number of elements in each set} \\ M_i &= |\bigcup_{j=1}^{i} S_j| \text{ the number of unique elements so far} \\ u_i &= M_i - M_{i-1} \text{ the number of new elements} \end{aligned}$$

- The likelihood function is:

$$L = \prod_{j=1}^{n} \binom{M_{j-1}}{k_j - u_j} \binom{m - M_{j-1}}{u_j} p^{k_j} (1-p)^{m-k_j} .$$

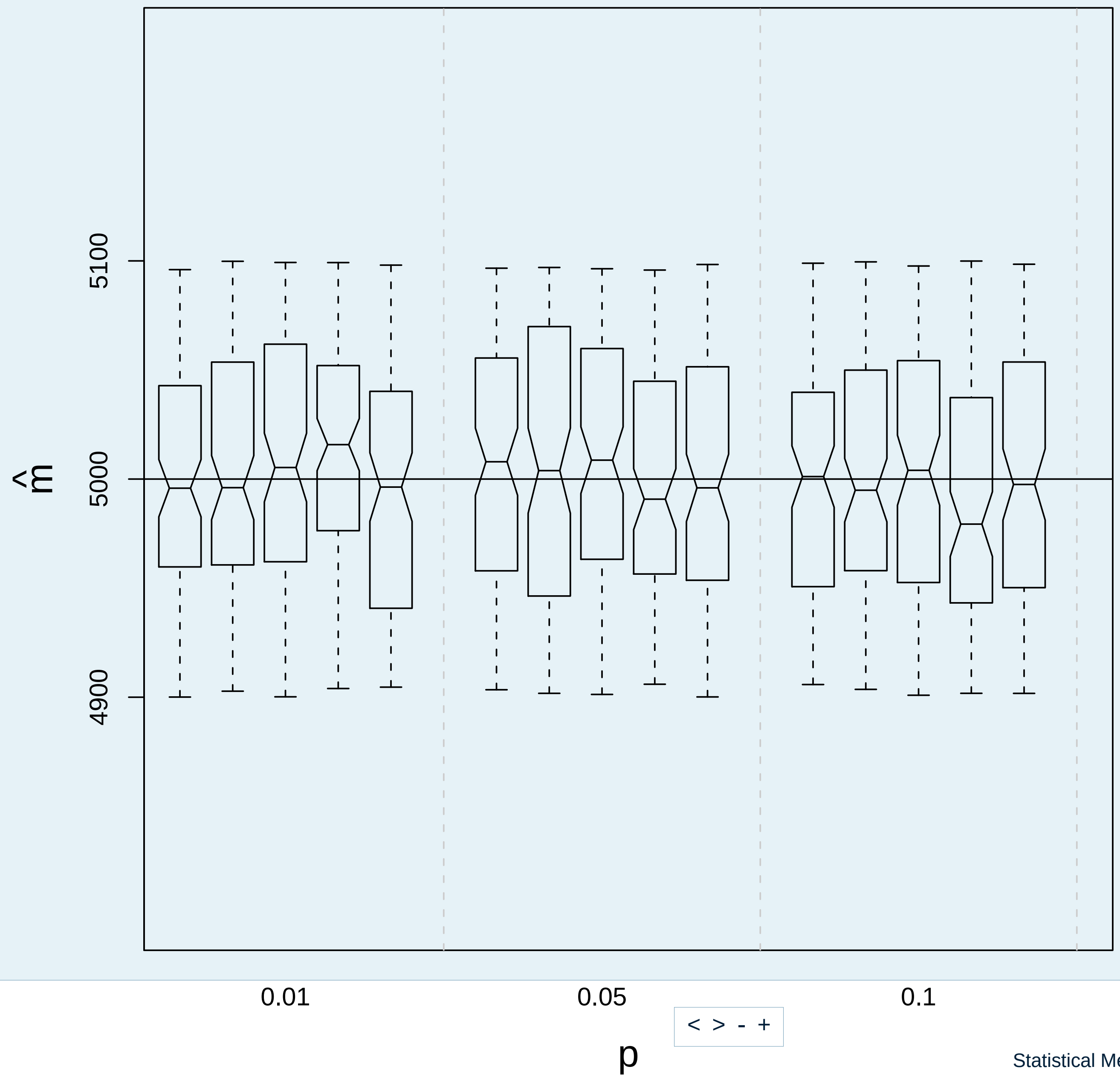
< > - +

Monte Carlo Results



< > - +

Monte Carlo Results

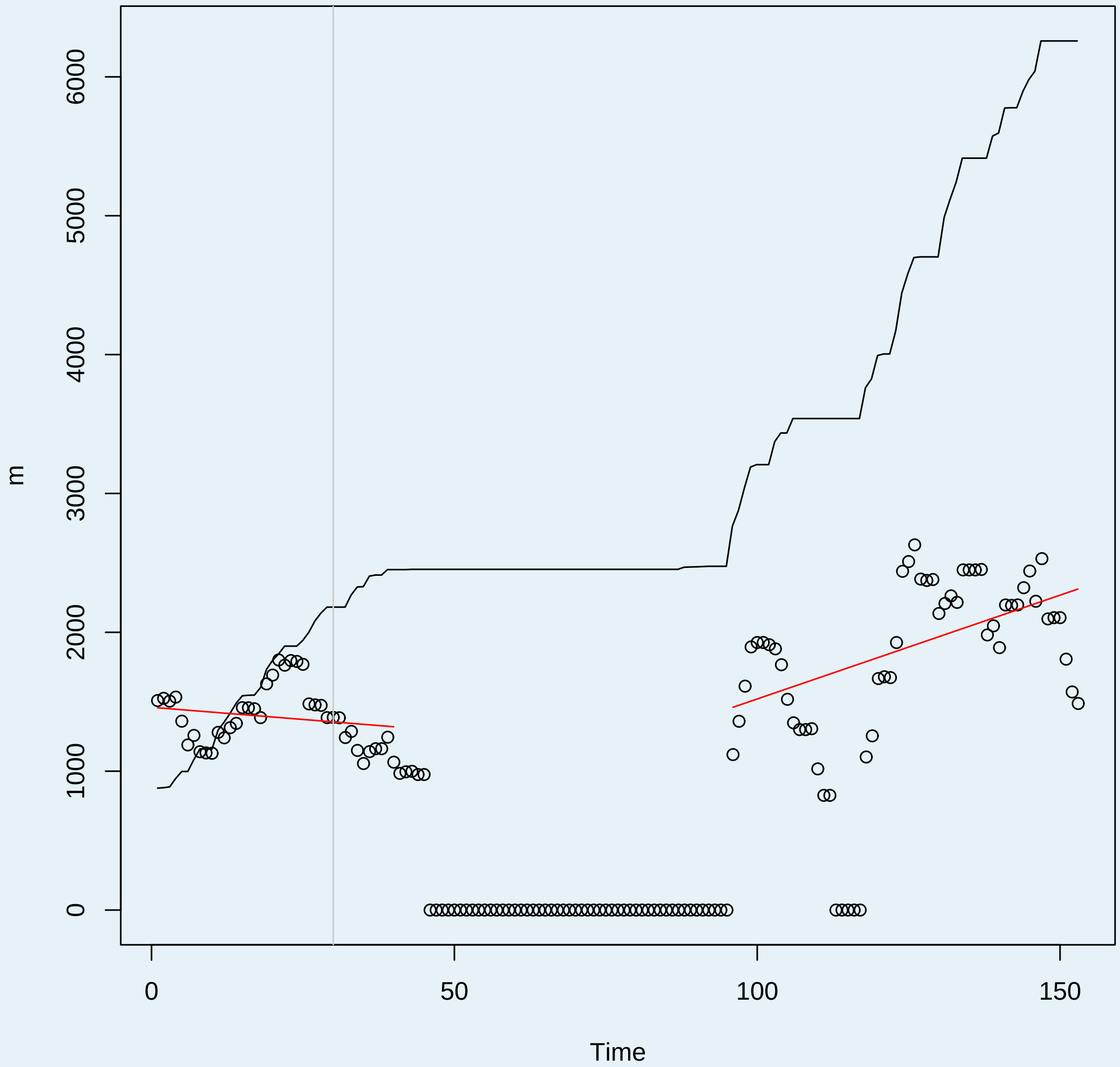


< > - +

Users and Network Traffic

- 41 user/host pairs tracked for 5 months. Noted web servers visited in this period.
- For each week and each user u_i the set S_i is the set of servers visited.
- Obtain a time series of intersection graphs.
- Some missing data.
- Considered graphs constructed on 1 week of data, with a 1 day increment.
- So each observation shares 6 day's worth of surfing resulting in a highly dependent timeseries.

Estimates: m

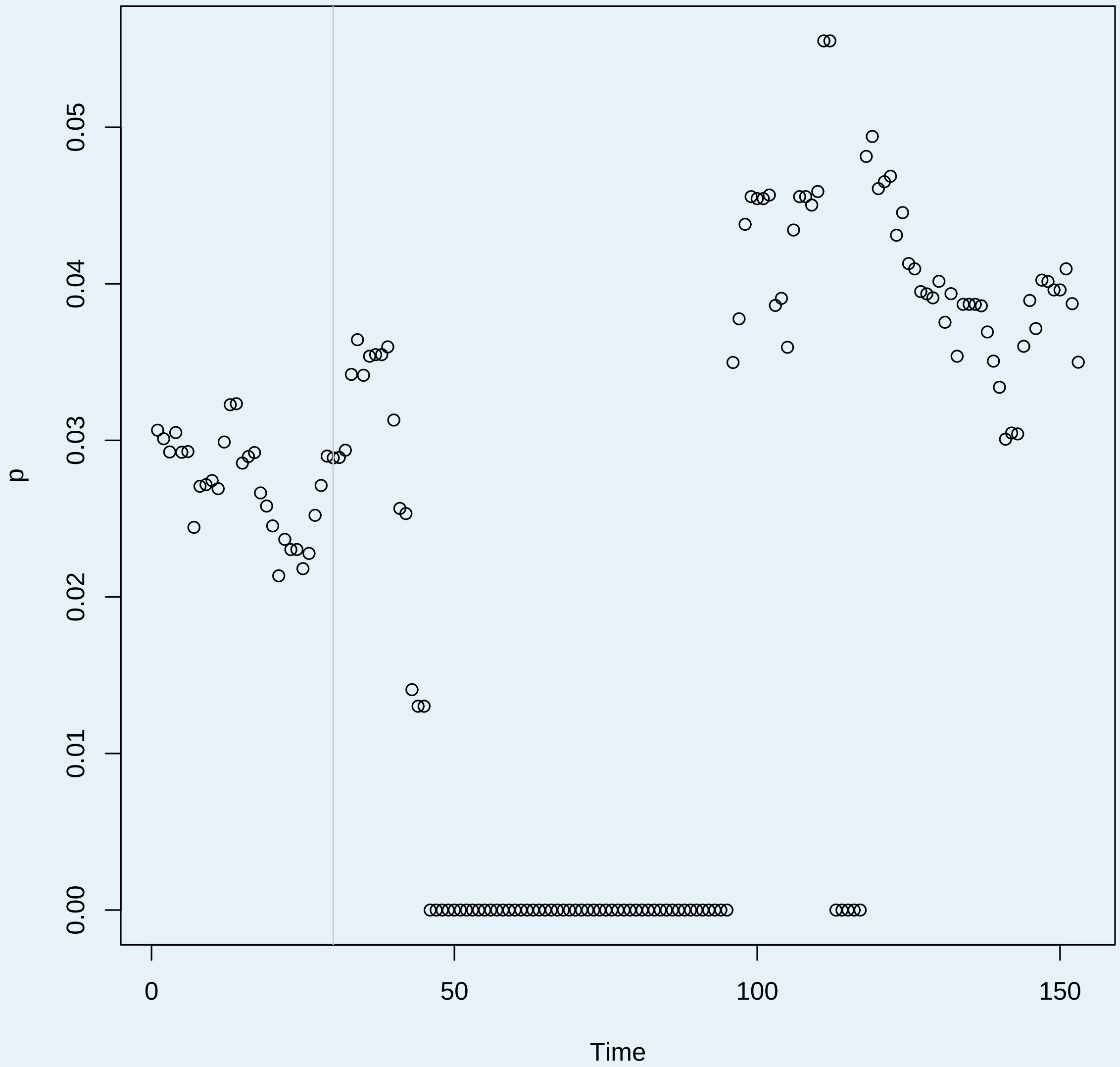


< > - +

Comments on m

- Before the end of the fiscal year users are drawing from around 1500 web servers.
- This appears quite stable within the short period investigated.
- The dependence caused by overlapping windows is obvious.
- In Nov/Dec the number of web servers appears to be increasing to 2000.
- It may have stablized.
- The actual pool of web servers is changing in time.

Estimates: p



Comments on p

- Similarly to m, p seems to have slightly different values before and after the beginning of the fiscal year.
- There seems to be more noise in the estimates of p than there were in m.

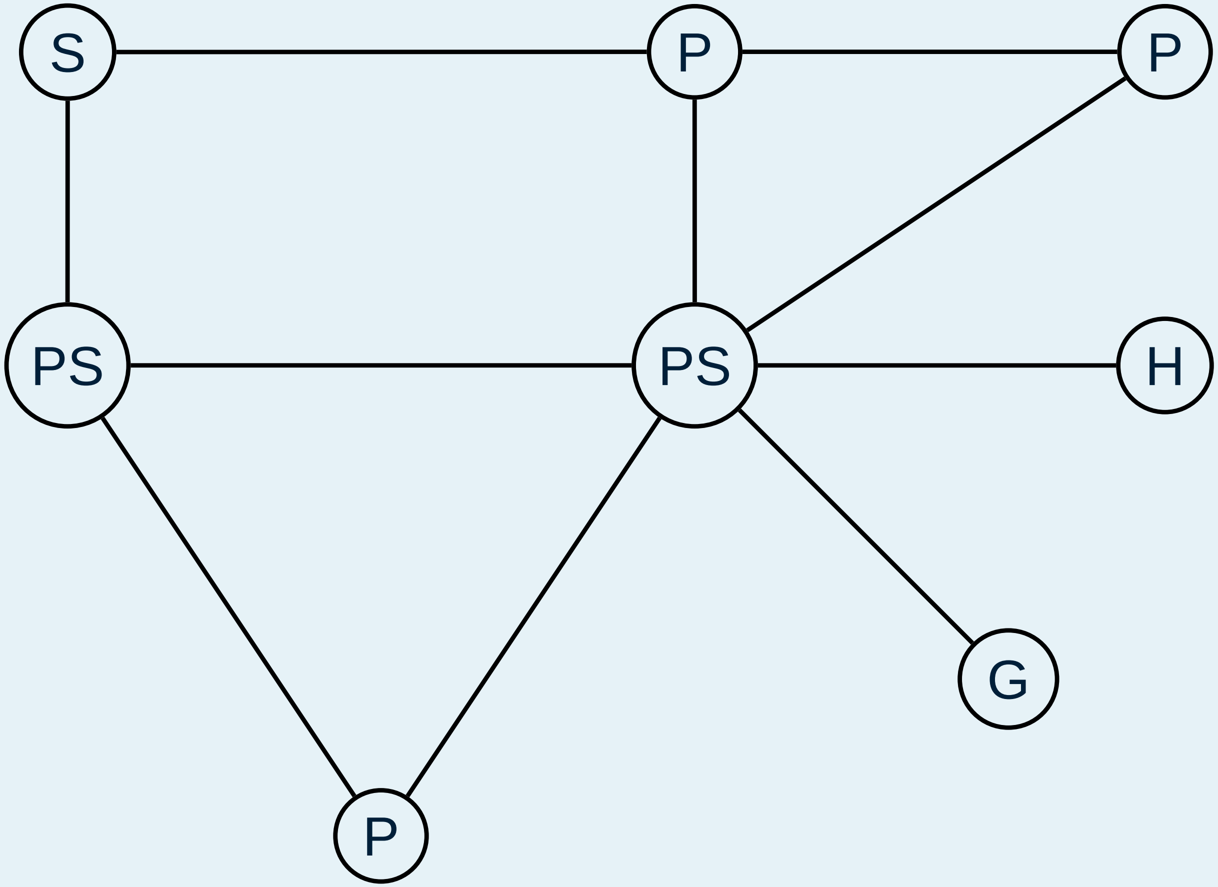
Significance Tests

- Are users visiting the same web servers?
- Consider the intersection of the servers visited by users A and B.
- Is this larger than expected?
- Hypergeometric distribution.
  - This allows us to test the significance of the size of the intersection.
- Multiple comparisons correction via Bonferroni (0.05 level).
  - Since we are making many tests, we need to correct for this.
- Looking at user visits collected over a one week period.

Significance Tests

- During this week, eight pairs of users visited more servers in common than would be expected (p-value < 0.001).
- Next we want to consider the time series of graphs.
- We obtain a time series of significant edges.
- Are these generally the same users?
- Consider overlapping one week windows, incremented by one day.
- Keep only the edges that appear in the most windows.

Significance Over Time: 95th quantile



< > - +

Conclusions

- Random intersection models provide a method for analyzing large-scale behavior of users.
- User behavior changes with time.
- Users tend to have more in common with each other than the model would suggest.
- This is an extremely simplistic model. Each user should (maybe?) have their own p. There should be more than one class of web sites (ubiquitous, common, rare?), so users would have several (three) probabilities.

< > - +

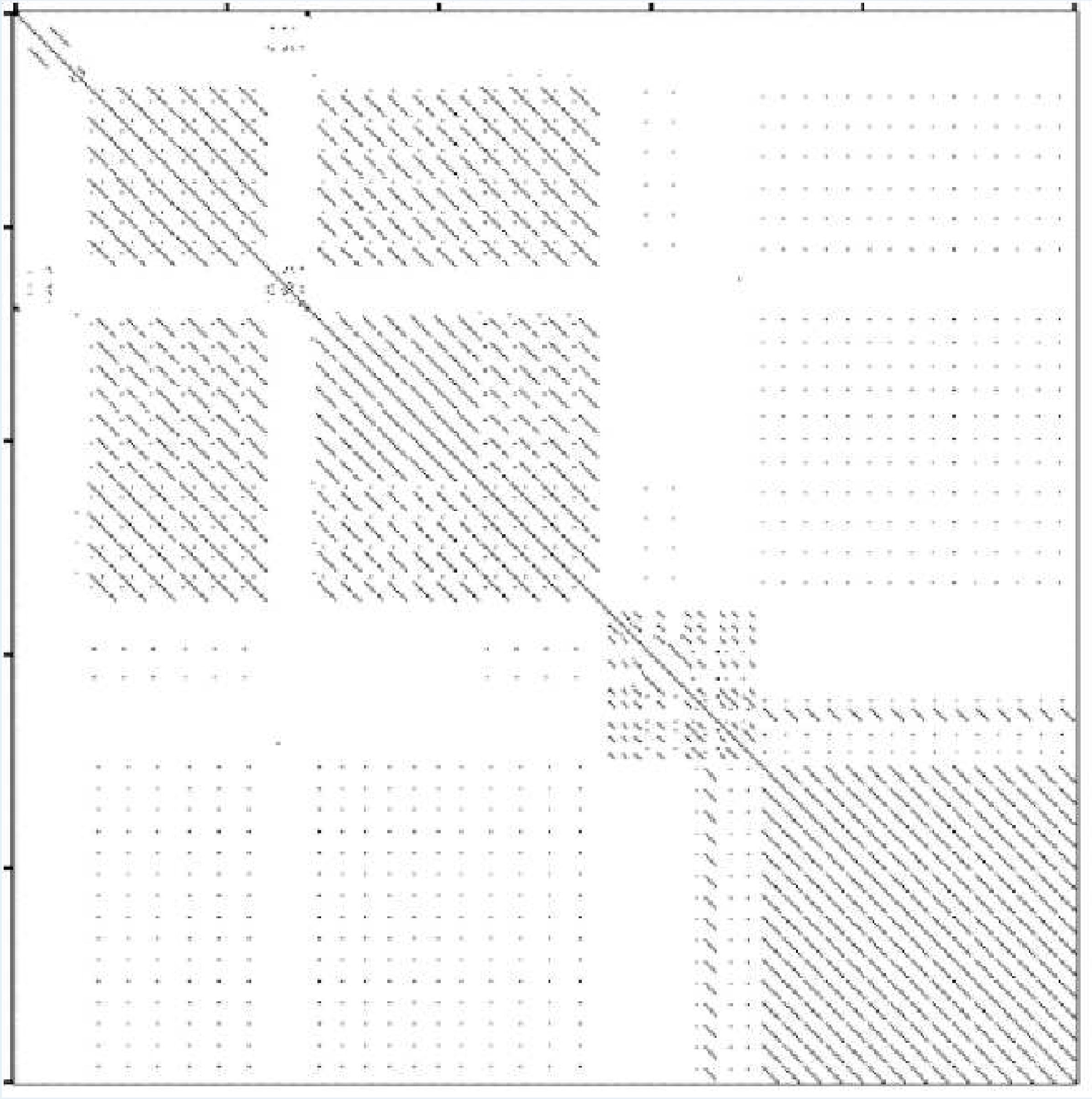
Profiling Users on a Host

- We want to be able to re-authenticate users on-the-fly.
- Given what the user is doing (typing, moving the mouse, executing programs) can we tell that the user is/is not the authorized user defined by the login?
- People have looked at keystroke timing, command lines, mouse movement, etc.
- In a windows environment, window titles take the place of command lines, so we investigated the utility for authenticating users based on the pattern of window titles.
- We used a simple intersection classifier (plus variations): how many titles are in common with previous logins (assumed to be authentic)?

Window Titles

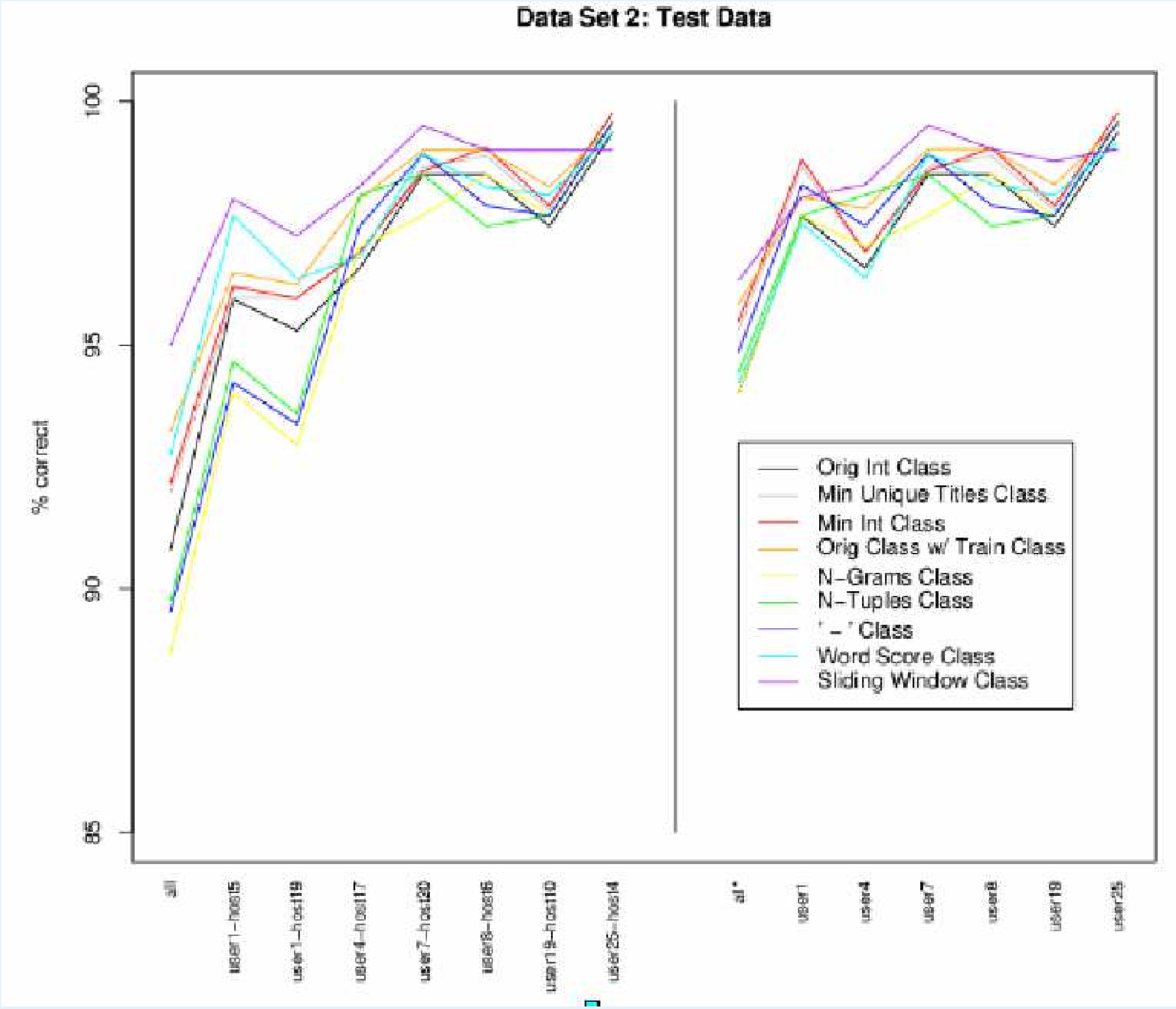
#	#Sessions	Users	Window
7002	425	1-19,1-5,19-10,25-4,4-17,7-20,8-6	Inbox - Microsoft Outlook
2525	411	1-19,1-5,19-10,25-4,4-17,7-20,8-6	Program Manager
2188	215	1-19,1-5,19-10,25-4,4-17,7-20,8-6	Microsoft Word
792	126	1-19,19-10,4-17,7-20,8-6	Netscape
704	156	1-19,1-5,19-10,25-4,4-17,7-20,8-6	Print
672	213	1-19,1-5,19-10,25-4,4-17,7-20,8-6	Microsoft Outlook
639	156	19-10,4-17,7-20,8-6	<<12761>> <<9227>>
592	170	1-19,1-5,19-10,25-4,4-17,7-20,8-6	<<16193>> - Message (<<16184>> <<5748>>)
555	174	1-19,1-5,19-10,25-4,4-17,7-20,8-6	<<6893>> <<13916>>
414	297	1-19,1-5,19-10,4-17,7-20,8-6	Microsoft(<<3142>>) Outlook(<<3142>>) <<7469>>
413	36	25-4	<<13683>> <<3653>> - Microsoft Internet Explorer
403	33	25-4	<<13683>> <<10676>> - Microsoft Internet Explorer
402	309	1-19,1-5,19-10,25-4,4-17,7-20,8-6	- Microsoft Outlook
401	61	1-19,1-5,19-10,25-4,4-17,7-20,8-6	Microsoft PowerPoint
		....	
198	84	1-19,1-5	http://<<1718>>.<<7267>>.<<4601>>/<<16345>>
125	22	25-4	http://<<9318>>.<<9500>>.<<3503>>.<<9193>>.<<4601>

Cross Plot



< > - +

Results



Data Set 2: Test Data
% correct
100
95
90
85
all
user1−host15
user1−host19
user4−host17
user7−host20
user8−host16
user19−host10
user25−host4
all *
user1
user4
user7
user8
user19
user25
Orig Int Class
Min Unique Titles Class
Min Int Class
Orig Class w/ Train Class
N−Grams Class
N−Tuples Class
' − ' Class
Word Score Class
Sliding Window Class

< > - +

The End...

- We have a large number of interest areas, and this has been a look at some of them. Other areas include:
  - Streaming data methods.
  - Profiling applications by their packet streams.
  - Visualization.
  - Text processing: cross-corpus discovery.
  - Computer virus propagation models.
  - Classifier research.
  - Integrating sensors and classifiers/clustering.
  - Manifold learning, metric geometry.