A Robust Classifier for Passive TCP/IP Fingerprinting

Rob Beverly
MIT CSAIL
rbeverly@csail.mit.edu

April 20, 2004

PAM 2004

Outline

- A Robust Classifier for Passive TCP/IP Fingerprinting
  - Background
  - Motivation
  - Our Approach/Description of Tool
  - Application 1: Measuring an Exchange Point
  - Application 2: NAT Inference
  - Conclusions
  - Questions?

Background

- Objective: Identify Properties of a Remote System over the Network
- Grand Vision: Passively Determine TCP Implementation in Real Time [Paxson 97]
- Easier: Identify Remote Operating System/Version Passively $\rightarrow$ “Fingerprinting”
- What’s the Motivation?

Motivation

- Fingerprinting Often Regarded as Security Attack
- Fingerprinting as a Tool:
  - In Packet Traces, Distinguish Effects due to OS from Network Path
  - Intrusion Detection Systems [Taleck 03]
  - Serving OS-Specific Content
- Fingerprinting a Section of the Network:
  - Provides a Unique Cross-Sectional View of Traffic
  - Building Representative Network Models
  - Inventory

Motivation Con’t

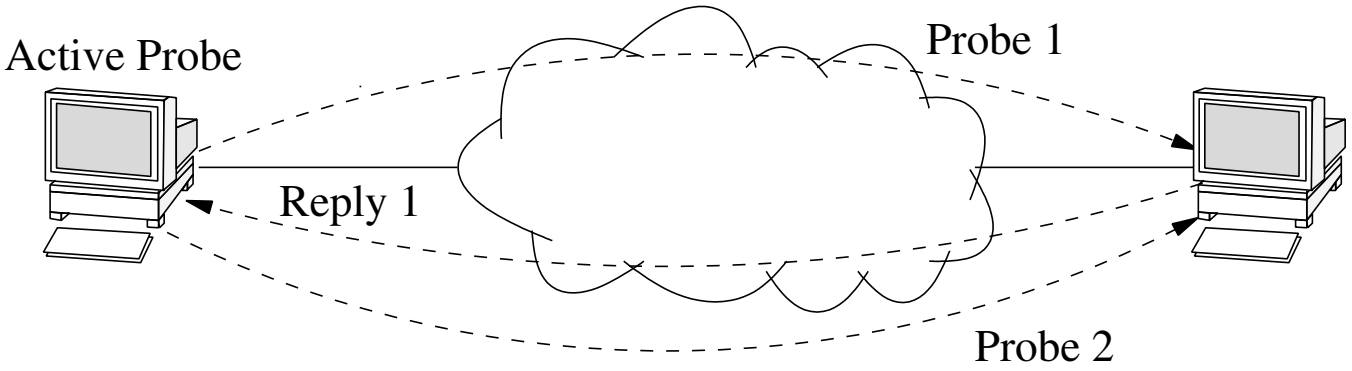
- We Select Two Applications:
  - Characterizing One-Hour of Traffic from Commercial Internet USA Exchange Point
  - Inferring NAT (Network Address Translation) Deployment
- More on these later...

TCP/IP Fingerprinting Background

- Observation: TCP Stacks between Vendors and OS Versions are Unique
- Differences Due to:
  - Features
  - Implementation
  - Settings, e.g. socket buffer sizes
- Two Ways to Fingerprint:
  - Active
  - Passive

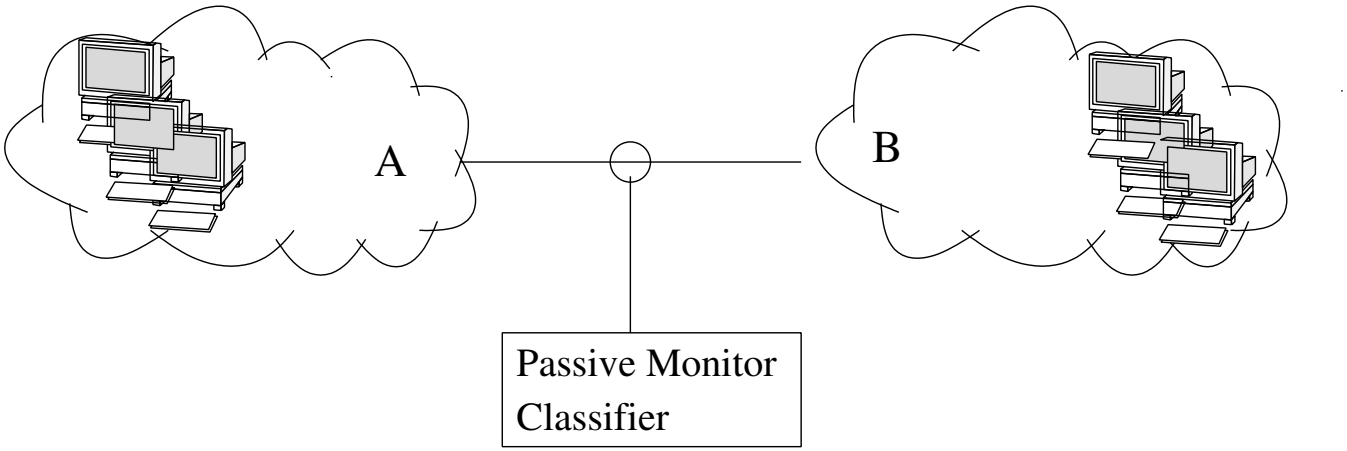
TCP/IP Fingerprinting Con't

- Active Fingerprinting
  - A "Probe" Host Sends Traffic to a Remote Machine
  - Scans for Open Ports
  - Sends Specially Crafted Packets
  - Observe Response; Match to list of Response Signatures.



TCP/IP Fingerprinting Con't

- Passive Fingerprinting
  - Assume Ability to Observe Traffic
  - Make Determination based on Normal Traffic Flow



Active vs. Passive Fingerprinting

- Active Fingerprinting
  - Advantages: Can be run anywhere, Adaptive
  - Disadvantages: Intrusive, detectable, not scalable
  - Tool: `nmap`. Database of $\sim$450 signatures.
- Passive Fingerprinting
  - Advantages: Non-intrusive, scalable
  - Disadvantages: Requires acceptable monitoring point
  - Tool: `p0f` relies on SYN uniqueness exclusively
- We want to Fingerprint all Traffic on a Busy, Representative Link
- Use Passive Fingerprinting

Robust Classifier

- Passive Rule-Based tools on Exchange Point Traces:
  - Fail to identify up to $\sim$ 5% of trace hosts
- Problems:
  - TCP Stack “Scrubbers” [Smart, et. al 00]
  - TCP Parameter Tuning
  - Signatures must be Updated Regularly
- Idea: Use Statistical Learning Methods to make a “Best-Guess” for each Host

Robust Classifier Con't

- Created Classifier Tool:
  - Naive Bayesian Classifier
  - Maximum-Likelihood Inference of Host OS
  - Each Classification has a Degree of Confidence
- Difficult Question: How to Train Classifier?
- Train Classifier Using:
  - `p0f` Signatures ($\sim$ 200)
  - Web-Logs
  - Special Collection Web Page + Altruistic Users

Robust Classifier Con't

- Question: Why not Measure OS Distribution using, e.g. Web Logs?
  - Want General Method, Not HTTP-Specific
  - Avoid Deep-Packet Inspection
  - Web Browsers Can Lie for anonymity and compatibility

Robust Classifier Con’t

- Inferences Made Based on Initial SYN of TCP Handshake
- Fields with Differentiation Power:
  - Originating TTL (0-255, as packet left host)
  - Initial TCP Window Size (bytes)
  - SYN Size (bytes)
  - Don’t Fragment Bit (on/off)

Robust Classifier Con't

- Originating TTL:
  - Next highest power of 2 trick
  - Example: Monitor Observes Packet with TTL=59. Infer TTL=64.
- Initial TCP Window Size can be:
  - Fixed
  - Function of MTU (Maximum Transmission Unit) or MSS (Maximum Segment Size)
  - Other
- Initial TCP Window Size Matching:
  - No visibility into TCP-options
  - For common MSS (1460, 1380, 1360, 796, 536) $\pm$ IP Options Size
  - Check if an Integer Multiple of Window Size

Example

Description	TTL	Win Size	SYN Size	DF	Conf	RuleBased Correct	Bayesian Correct
FreeBSD 5.2	64	65535	60	T	0.988	Y	Y
FreeBSD (1)	64	65535	70	T	0.940	N	Y
FreeBSD (2)	64	65530	60	T	0.497	N	Y

- Example 2: Tuned FreeBSD; Window Scaling Throws Off Ruled-Based

kern.ipc.maxsockbuf=4194304
net.inet.tcp.sendspace=1048576
net.inet.tcp.recvspace=1048576
net.inet.tcp.rfc3042=1
net.inet.tcp.rfc3390=1

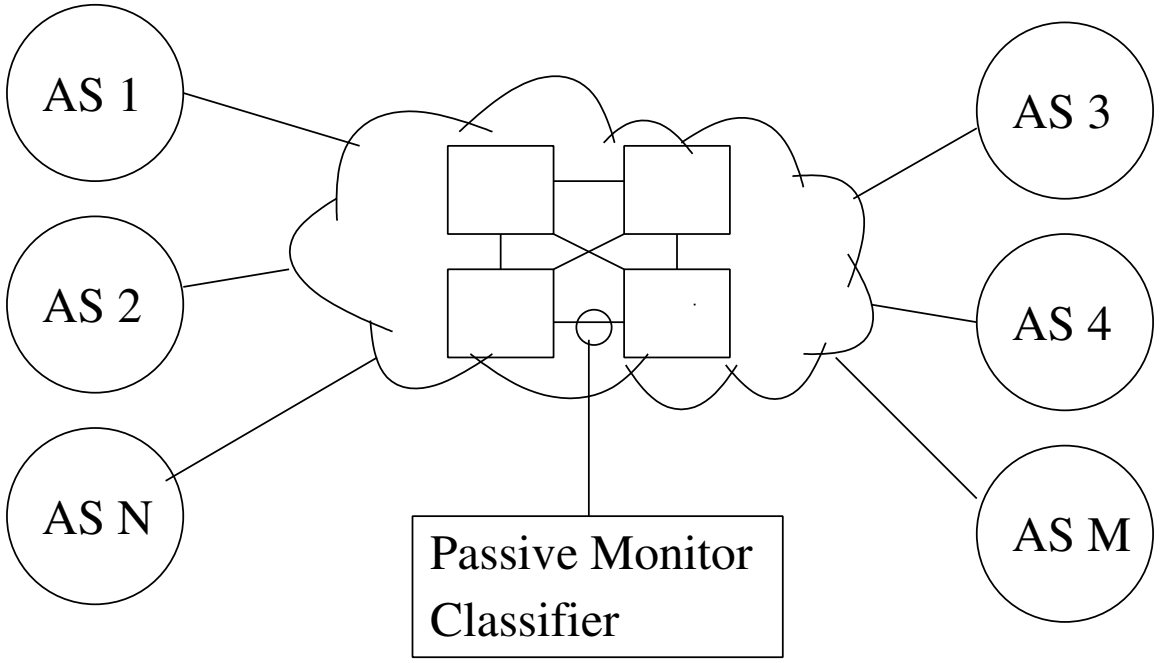
More Fields in Rule-based Approach → Fragile
Learning on Additional Fields → more Robust

Classifying a Cross-Section of the Internet

- Traces:
  - MIT LCS Border Router
  - NLANR MOAT
  - Commercial Internet Exchange Point Link (USA)
- Analyze One-Hour Trace from Exchange Point
- Collected in 2003 at 16:00 PST on a Wednesday

Classifying a Cross-Section of the Internet

- Traces:
  - Commercial Internet Exchange Point Link (USA)

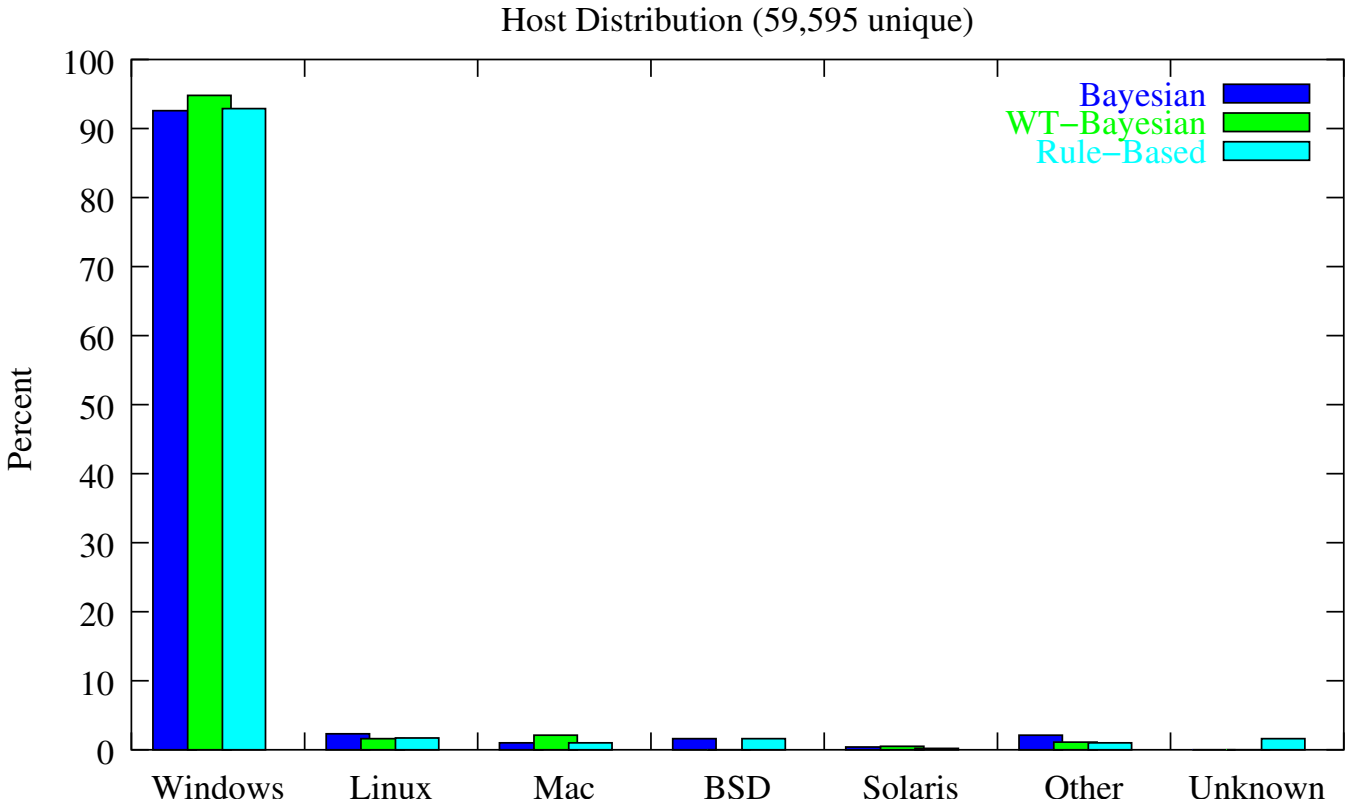


Classifying a Cross-Section of the Internet

- For Brevity (and Easier Computationally)
  - Group in Six Broad OS Categories
  - Measure Host, Packet and Byte Distribution
  - Using `p0f`-trained Bayesian, Web-trained Bayesian and Rule-Based

Host Distribution

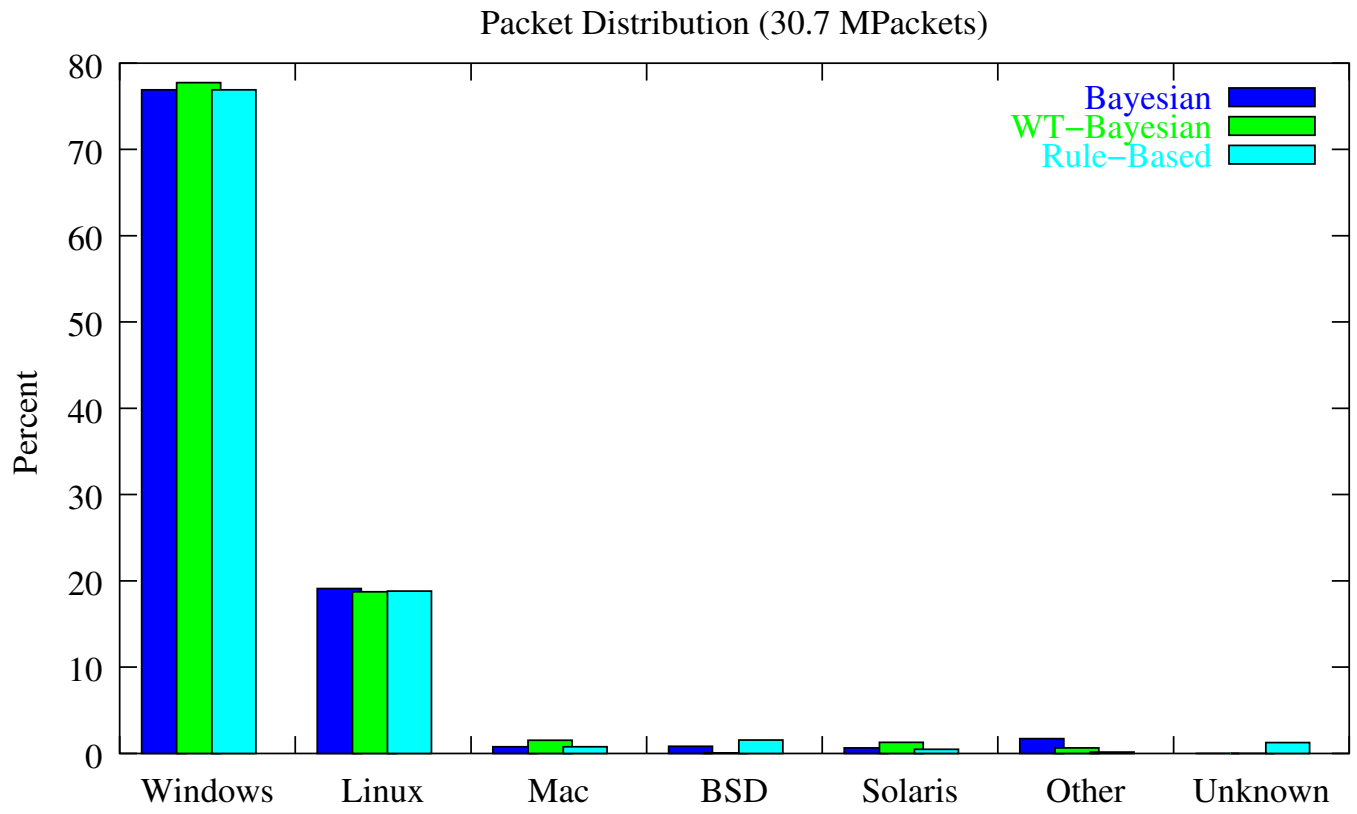
Windows Dominates Host Count: 92.6-94.8%



Note: Unknown applies only to Rule-Based

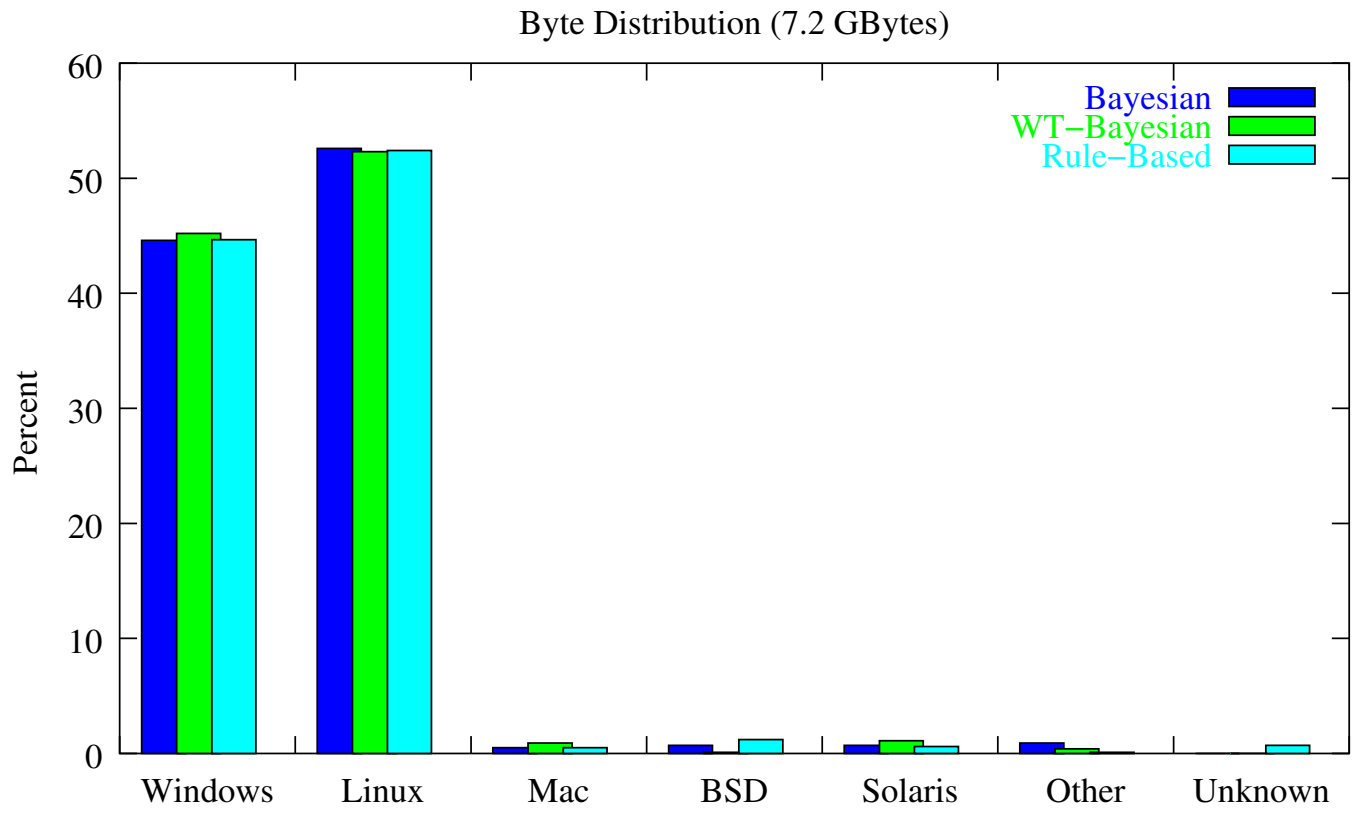
Packet Distribution

- Windows: 76.9-77.8%; Linux: 18.7-19.1%



Byte Distribution

- Windows: 44.6-45.2%; Linux: 52.3-52.6%



Byte Distribution

- Interesting Results
- Windows Dominates Hosts, but Linux hosts contribute the most traffic!
- Top 10 Largest Flows:
  - 55% of byte traffic!
  - 5 Linux, 2 Windows
  - Software Mirror, Web Crawlers (packet every 2-3ms)
  - SMTP servers
  - Aggressive pre-fetching web caches
- Conclusion: Linux Dominates Traffic, Primarily due to Server Applications in our Traces (YMMV)

Classifying for NAT Inference

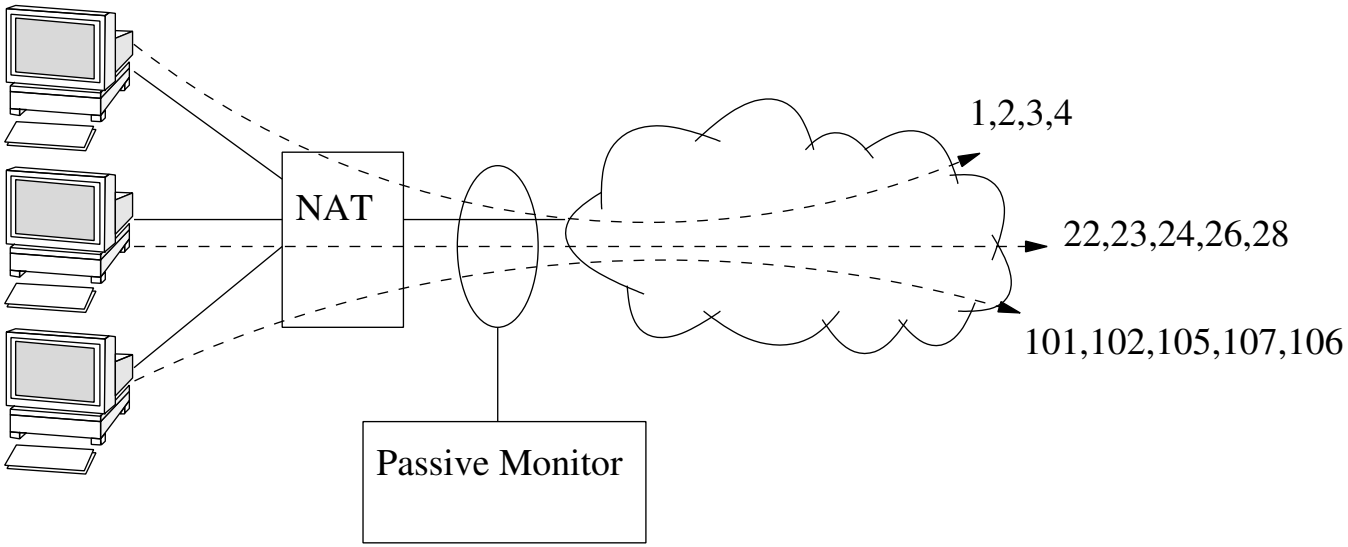
- Second Potential Application of Classifier
- Goal: Understand NAT prevalence in Internet
- Motivation: “E2E-ness” of Internet
- Assume hosts behind an IP-Masquerading NAT have different OS or OS versions (strong assumption)
- Look for traffic from same IP with different signature to get NAT lower-bound
- In hour-long trace, assume DHCP and dual-booting machine influence negligible

NAT Inference

- Existing Approaches: sflow [Phaal 03], IP ID [Bellovin 02]
- sflow:
  - Monitor must be before 1st hop router
  - Using TTL trick, look for unexpectedly low TTLs (decremented by NAT)

NAT Inference

- IP ID [Bellovin 02]:
  - If IP ID is a sequential counter
  - Construct IP ID sequences
  - Coalesce, prune with empirical thresholds
  - Number of remaining sequences estimates number of hosts



Sequence Matching Obstacles

- Question of whether IP ID Sequence Matching Works:
  - IP ID used for Reassembling Fragmented IP packets
  - No defined semantic, e.g *BSD uses pseudo-random number generator!
  - If DF-bit set, no need for reassembly. NAT sets IP ID to 0.
  - Proper NAT should rewrite IP ID to ensure uniqueness!
- Further, these obstacles will become significant in the future!
- We seek to determine the practical impact of these limitations and how well alternate approach works in comparison.

Evaluating NAT Inference Algorithms

- To evaluate different NAT inference algorithms
- Gathered $\sim$ 2.5M packets from academic building (no NAT)
- Synthesize NAT traffic
- Reduce number of unique addresses by combining traffic of n IP addresses into 1.
- We term n the “NAT Inflation” factor

Evaluating NAT Inference

- Synthetic Traces Created with 2.0 NAT Inflation Factor
- Inferred NAT Inflation:
  - IP ID Sequence Matching: 2.07
  - TCP Signature: 1.22
- IP ID Technique works well!
- TCP Classification does not have enough Discrimination Power

NAT Inflation in the Internet

- Results:
  - IP ID Sequence Matching: 1.092
  - TCP Signature: 1.02
- Measurement-based lower bound to understanding NAT prevalence in Internet

Future

- How to Validate Performance of Classifier? (What's the *Correct* Answer?)
- Expand Learning to Additional Fields/Properties of Flow
- Properly Train Classifier?
- Web Page (Honest Users Please!): http://momo.lcs.mit.edu/finger/finger.php
- Identifying TCP Stack Variant (e.g. Reno, Tahoe)

Conclusions

- Contributions of this Work:
  - Developed Robust tool for TCP/IP Fingerprinting
  - Measure Operating System host, packet and byte distribution “in the wild”
  - Understand NAT inflation factor
  - Measured $\sim$ 9% NAT inflation

Questions?

- Questions?
- More: http://momo.lcs.mit.edu/finger/finger.php
- Thanks!