Traffic Monitoring and Application Classification: A Novel Approach

Michalis Faloutsos, UC Riverside

Thomas Karagiannis   Marios Iliofotou
General Problem Definition

We don’t know what goes on in the network

- Measure and monitor:
  - Who uses the network? For what?
  - How much file-sharing is there?
  - Can we observe any trends?

- Security questions:
  - Have we been infected by a virus?
  - Is someone scanning our network?
  - Am I attacking others?
Problem in More Detail

- Given network traffic in terms of flows
  - **Flow**: tuple (source IP, port; dest IP, port; protocol)
  - Flow statistics: packet sizes, interarrival etc
- Find which application generates each flow
  - Or which flows are P2P
  - Or detect viruses/worms
- Issues:
  - Definition of flow hides subtleties
  - Monitoring tools, netflow, provide this
State of the Art Approaches

- **Port-based**: some apps use the same port
  - Works well for legacy applications, but not for new apps

- **Statistics-based methods**:
  - Measure packet and flow properties
    - Packet size, packet interarrival time etc
    - Number of packets per flow etc
  - Create a profile and classify accordingly
  - Weakness: Statistical properties can be manipulated

- **Packet payload based**:
  - Match the signature of the application in payload
  - Weakness
    - Require capturing the packet load (expensive)
    - Identifying the “signature” is not always easy

- **IP blacklist/whitelist filtering**
Our Novelty, Oversimplified

- We capture the intrinsic behavior of a user
  - Who talks to whom
- Benefits:
  - Provides novel insight
  - Is more difficult to fake
  - Captures intuitively explainable patterns
- Claim: our approach can give rise to a new family of tools
How our work differs from others

- **Previous work**
  - BLINC: Profile behavior of user (host level)
  - TDGs: Profile behavior of the whole network (network level)

- **Our work**
Motivation: People Really Care

- We started by measuring P2P traffic
  - which explicitly tries to hide
  - Karagiannis (UCR) at CAIDA, summer 2003
- How much P2P traffic is out there?
  - RIAA claimed a drop in 2003
  - We found a slight increase
    - "Is P2P dying or just hiding?" Globecom 2004
The Reactions

- RIAA did not like it
  - Respectfully said that we don’t know what we are doing
- The P2P community loved it
  - Without careful scrutiny of our method
More People Got Interested

- Wired: ```Song-Swap Networks Still Humming``` on Karagiannis' work.
- ACM news, PC Magazine, USA Today,...
- Congressional Internet Caucus (J. Kerry!)
- In litigation docs as supporting evidence!
Structure of the talk

- **Part I:**
  - BLINC: A host-based approach for traffic classification

- **Part II:**
  - Monitoring using the network-wide behavior: Traffic Dispersion Graphs, TDGs
Part I: BLINC Traffic classification

- The goal:
  - Classify Internet traffic flows according to the applications that generate them

- Not as easy as it sounds:
  - Traffic profiling based on TCP/UDP ports
    - Misleading
  - Payload-based classification
    - Practically infeasible (privacy, space)
  - Can require specialized hardware

Joint Work with: Thomas Karagiannis, UC Riverside/ Microsoft
Konstantina Papagiannaki, Nina Taft, Intel
The State of the Art

- Recent research approaches
  - Statistical/machine-learning based classification
    - Roughan et al., IMC’04
    - McGregor et al., PAM’05
    - Moore et al., SIGMETRICS’05
  - Signature based
    - Varghese, Fingerhut, Bonomi, SIGCOMM’06
    - Bonomi, et al. SIGCOMM’06
  - IP blacklist/whitelist filtering to block bad traffic
    - Soldo+, Markopoulou, ITA’08
  - UCR/CAIDA a systematic study in progress:
    - What works, under which conditions, why?
Our contribution: BLINC

- BLINd Classification
  - ie without using payload
- We present a fundamentally different “in the dark” approach
  - We shift the focus to the host
- We identify “signature” communication patterns
  - Difficult to fake
BLINC overview

- Characterize the host
  - Insensitive to network dynamics (wire speed)
- Deployable: Operates on flow records
  - Input from existing equipment
- Three levels of classification
  - Social: Popularity
  - Functional: Consumer/provider of services
  - Application: Transport layer interactions
Social Level

- **Social:**
  - Popularity
  - Bipartite cliques

- **Gaming communities identified by using data mining:**
  - fully automated cross-association
  - Chakrabarti et al KDD 20 (C. Faloutsos CMU)
Functional level

- Functional:
  - Infer role of node
    - Server
    - Client
    - Collaborator
  - One way: #source ports vs. # of flows
Social level

- Characterization of the popularity of hosts
- Two ways to examine the behavior:
  - Based on number of destination IPs
  - Analyzing communities
Social level: Identifying Communities

- Find bipartite cliques
Social Level: What can we see

- Perfect bipartite cliques
  - Attacks
- Partial bipartite cliques
  - Collaborative applications (p2p, games)
- Partial bipartite cliques with same domain IPs
  - Server farms (e.g., web, dns, mail)
Social Level:
Finding communities in practice

- Gaming communities identified by using data mining: fully automated cross-association
  Chakrabarti et al KDD 2004 (C. Faloutsos CMU)
Functional level

- Characterization based on tuple (IP, Port)
- Three types of behavior
  - Client
  - Server
  - Collaborative
Functional level: Characterizing the host

$Y$-axis: number of source ports  $X$-axis: number of flows

Collaborative applications: No distinction between servers and clients

Obscure behavior due to multiple mail protocols and passive ftp
Application level

- Interactions between network hosts display diverse patterns across application types.

- We capture patterns using *graphlets*:
  - Most typical behavior
  - Relationship between fields of the 5-tuple
Application level: Graphlets

- Capture the behavior of a single host (IP address)
- Graphlets are graphs with four "columns":
  - src IP, dst IP, src port and dst port
- Each node is a distinct entry for each column
  - E.g. destination port 445
- Lines connect nodes that appear on the same flow
### Graphlet Generation (FTP)

<table>
<thead>
<tr>
<th>sourceIP</th>
<th>destinationIP</th>
<th>sourcePort</th>
<th>destinationPort</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Z</td>
<td>21</td>
<td>3000</td>
</tr>
<tr>
<td>X</td>
<td>Z</td>
<td>1026</td>
<td>3001</td>
</tr>
<tr>
<td>X</td>
<td>U</td>
<td>21</td>
<td>5000</td>
</tr>
<tr>
<td>X</td>
<td>U</td>
<td>20</td>
<td>5005</td>
</tr>
</tbody>
</table>

![Graph representation](image)
What can Graphlets do for us?

- **Graphlets**
  - are a compact way to profile of a host
  - capture the intrinsic behavior of a host

- **Premise:**
  - Hosts that do the same, have similar graphlets

- **Approach**
  - Create graphlet profiles
  - Classify new hosts if they match existing graphlets
Training Part:
Create a Graphlet Library

(a) ATTACK
(b) ATTACK
(c) ATTACK
(d) WEB (TCP)
(e) GAMES (UDP)
(f) CHAT (TCP)
(g) P2P (TCP, UDP)
(h) GAMES/UDP
(i) DNS/UDP
(j) FTP
(k) MAIL
(l) STREAMING/REAL
(m) MAIL server with DNS
Additional Heuristics

- In comparing graphlets, we can use other info:
  - the transport layer protocol (UDP or TCP).
  - the relative cardinality of sets.
  - the communities structure:
    - If X and Y talk to the same hosts, X and Y may be similar
    - Follow this recursively

- Other heuristics:
  - Using the per-flow average packet size
  - Recursive (mail/dns servers talk to mail/dns servers, etc.)
  - Failed flows (malware, p2p)
Evaluating BLINC

- We use real network traces
- Data provided by Intel:
  - Residential (Web, p2p)
  - Genome campus (ftp)
- Train BLINC on a small part of the trace
- Apply BLINC on the rest of the trace
Compare with what?

- Develop a reference point
  - Collect and analyze the whole packet
  - Classification based on payload signatures
- Not perfect but nothing better than this
Classification Results

- **Metrics**
  - **Completeness**
    - Percentage classified by BLINC relative to benchmark
    - “Do we classify most traffic?”
  - **Accuracy**
    - Percentage classified by BLINC correctly
    - “When we classify something, is it correct?”
  - **Exclude unknown and nonpayload flows**
Classification results: Totals

- BLINC works well

80%-90% completeness!
>90% accuracy!!
Characterizing the unknown: Non-payload flows

BLINC is not limited by non-payload flows or unknown signatures

Flows classified as attacks reveal known exploits
BLINC issues and limitations

- How do we compare graphlets?
  - “Graph similarity” is difficult to define
  - Currently, based on heuristics and training
- What if a node runs two apps at the same time?
- Extensibility
  - Creating and incorporating new graphlets
- Application sub-types
  - e.g., BitTorrent vs. Kazaa
- Access vs. Backbone networks?
  - Works better for access networks (e.g. campus)
Developing a Useable Tool

- Java front-end by Dhiman Barman UCR
Follow up work:
Profiling the end user

- We examine the dynamics of profiling
- How much variability exists
  - Per node over time
  - Among nodes in a network
- How can I summarize a graphlet
  - So that I can compare it with others?
- The answers in PAM 2007
Conclusions - I

- We shift the focus from flows to hosts
  - Capture the intrinsic behavior of a host
- Multi-level analysis:
  - each level provides more detail
- Good results in practice:
  - BLINC classifies 80-90% of the traffic with greater than 90% accuracy
Part II: Traffic Dispersion Graphs

- Monitoring traffic as a network-wide phenomenon

Paper at Internet Measurement Conference (IMC) 2007
Joint work with: Marios Iliofotou UC Riverside, G. Varghese UCSD
Prashanth Pappu, Sumeet Singh (Cisco) M. Mitzenmacher (Harvard)
Traffic Dispersion Graphs:

- Who talks to whom
- Deceptively simple definition
- Provides powerful visualization and novel insight
Defining TDGs

- A node is an IP address (host, user)
- A key issue: define an edge (Edge filter)
  - Edge can represent different communications
  - Simplest: edge = the exchange of any packet
  - Edge Filter can be more involved:
    - A number of pkts exchanged
    - TCP with SYN flag set (initiating a TCP connection)
    - sequence of packets (e.g., TCP 3-way handshake)
    - Payload properties such as a content signature
Generating a TDG

- Pick a monitoring point (router, backbone link)
- Select an edge filter
  - Edge Filter = “What constitutes an edge in the graph?”
  - E.g., TCP SYN Dst. Port 80
- If a packet satisfies the edge filter, create the link
  - srcIP → dstIP
- Gather all the links and generate a graph
  - within a time interval, e.g., 300 seconds (5 minutes)
TDGs are a New Kind of Beast

- TDGs are
  - Directed graphs
  - Time evolving
  - Possibly disconnected
- TDGs are not yet another scalefree graph
- TDGs are not a single family of graphs
  - TDGs with different edge filters are different
- TDGs hide a wealth of information
  - Make “cool” visualizations
  - Can be “mined” to provide novel insight
TDGs and Preliminary Results

- We focus on studying **port-based TDGs**
  - Even that can give interesting information
- We study destination ports of known applications:
  - **UDP** ports: we generate an edge based on the first packet between two hosts
  - **TCP** we add an edge on a TCP SYN packet for the corresponding destination port number
    - e.g., port 80 for HTTP, port 25 for SMTP etc.
Data Used

- Real Data: typical duration = 1 hour
  - OC48 from CAIDA (22 million flows, 3.5 million IPs)
  - Abilene Backbone (23.5 million flows, 6 million IPs)
  - WIDE Backbone (5 million flows, 1 million IPs)
  - Access links traces (University of Auckland) + UCR traces were studied but not shown here (future work)
TDGs as a Visualization Tool
Identifying Hierarchies

- Hierarchical structure with multiple levels of hierarchy

M. Faloutsos UCR
Web Traffic

Web: https

Web: port 8080
TDG Visualizations (Peer-to-Peer)

WinMX P2P App
- UDP Dst. Port 6257
- 15 sec

Observations
- Many nodes with in-and-out degree (InO)
- One large connected component
- Long chains

Zoom

InO degree
Bidirectional
Detecting Viruses and Unusual Activities

Random IP range scanning activity?

Slammer: port 1434

NetBIOS: port 137
Visually detecting virus activity

- **Virus (slammer) creates more “star” configurations**
- **Directivity makes it clearer**
  - Center node -> nodes, for virus “stars”
Quantitative Study of TDGs
Using Graph Metrics

- We use **new** and commonly used metrics
- Degree distribution
- Giant Connected Component
  - Largest connected subgraph
- Number of connected components
- In-Out nodes
  - Node with in- and out- edges
- Joint Degree Distribution
The degree distributions of TDGs varies a lot.

- Only some distributions can be modeled by power-laws (HTTP, DNS).
- P2P communities (eDonkey) have many medium degree nodes (4 to 30).
- HTTP and DNS have few nodes with very high degrees.
- NetBIOS: Scanning activity: 98% of nodes have degree of one, few nodes with very high degree → scanners
Joint Degree Distribution (JDD)

- **JDD**: $P(k_1, k_2)$, the probability that a randomly selected edge connects nodes of degrees $k_1$ and $k_2$
  - Normalized by the total Number of links

M. Faloutsos UCR
Joint Degree Distribution (JDD)

- Couture plots (log-log scale due to high variability)
  - x-axis: Degree of the node on the one end of the link
  - y-axis: Degree of the other node
- Observations:
  - HTTP: low degree client to low to high degree servers
  - WinMX: medium degree nodes are connected
  - DNS: signs of both client server and peer-to-peer behavior
- Top degree nodes are not directly connected (top right corner)
TDGs Can Distinguish Applications

- Monitor the top 10 ports number in number of flows.
- Scatter Plot:
  - Size of GCC Vs number of connected components.
  
  **Stable over Time!**

- We can separate apps!

  - **Soribada**
    - UDP port 22321
    - UDP port 7674
  
  - **WinMX**
    - UDP port 6257
  
  - **eDonkey**
    - TCP port 4662
    - UDP port 4665
  
  - **NetBIOS**
    - UDP port 137
  
  - **MS-SQL-S**
    - TCP port 1433

M. Faloutsos UCR
TDGs as a Monitoring/Security Tool

- Two modes of operation:
  - Classification: based on previously observed thresholds.
  - Security: calculate TDGs and trigger an alarm on large change

- How do we choose which TDGs to monitor?
  - Manually,
  - Automatically-adaptively,
  - Using automatically extracted signatures of content (Earlybird)
Final Conclusions

- The “behavior” of hosts hides information
  - Studying the transport-layer can provide insight
- We can do this at two levels
  - Host level using BLINC
  - Network-wide level using TDGs
- Advantages:
  - More difficult to fake
  - More intuitive to interpret and deploy
- It can be used to monitor and secure
My Areas of Research

- Measurements and models for the Internet
  - Network Topology: models and patterns [ToN03, CSB06, NSDI07]
  - Traffic monitoring: models and classification [sigcomm05] [PAM07]
- Routing Security
  - Modeling and Securing BGP routing NEMECIS: [Infocom04, 07]
  - Adhoc routing security: [ICNP 06][ICNP07]
- Quantifying and protecting against URL hijacking [miniInfocom08]
- Design and capacity of WLANs and hybrid nets [mobicom07, infocom08]
- DART: A radical network layer for ad hoc [Infocom04] [ToN06]
- Cooperative Diversity in ad hoc networks [JSAC06, Infocom06]
Extras
Main research areas

- Measurements
  - Traffic, BGP routing and topology, ad hoc
- Routing
  - scalable ad hoc, BGP instability
- Security
  - DoS, BGP attacks, ad hoc DoS
- Designing the future network
  - Rethinking the network architecture
**TDG Visualization (DNS)**

**DNS TDG**
- UDP Dst. Port 53
- 5 seconds

In- and Out-degree nodes

Very common in DNS, presence of few very high degree node

One large Connected Component!
(even in such small interval)
**Observations**

- There is **not** a large connected component as in DNS
- Clear roles
  - very few nodes with in- and-out degrees
    - Web caches?
    - Web proxies?
- Many disconnected components
TDG Visualization (Slammer Worm)

**Slammer Worm**
- UDP Dst. port 1434
- 10 seconds
- About:
    - Trace: April 24th
  - Observations (Scanning Activity)
    - Many high out-degree nodes
    - Many disconnected components
    - The majority of nodes have **only in-degree** (nodes being scanned)