Detecting Stealthy P2P Botnets Using Statistical Traffic Fingerprints

Junjie Zhang[1], Roberto Perdisci[2], Wenke Lee[1],
Unum Sarfraz[1], and Xiapu Luo[3]

[1]Georgia Tech [2]Univ. of Georgia
[3]Hong Kong Polytechnic Univ.

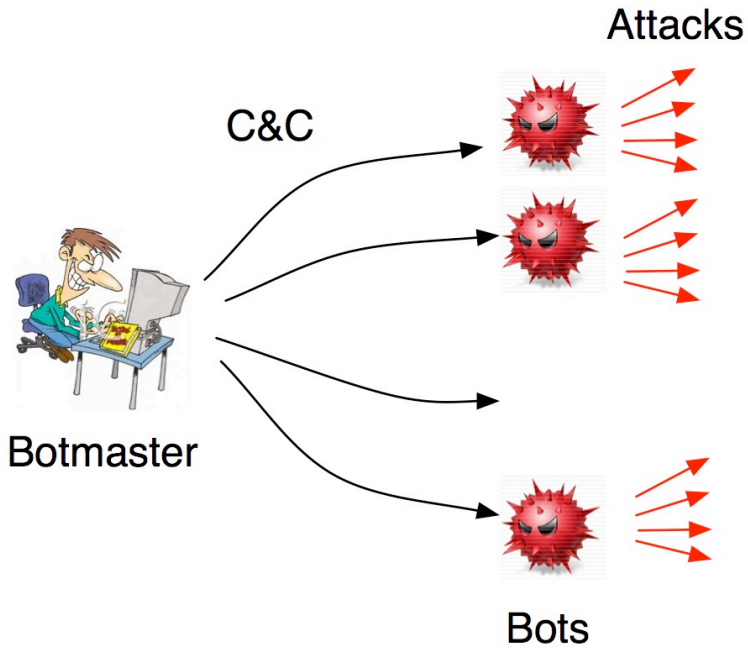
Georgia Tech College of Computing

Outline

- Introduction
- System Design
- System Evaluation
- Conclusion

Introduction

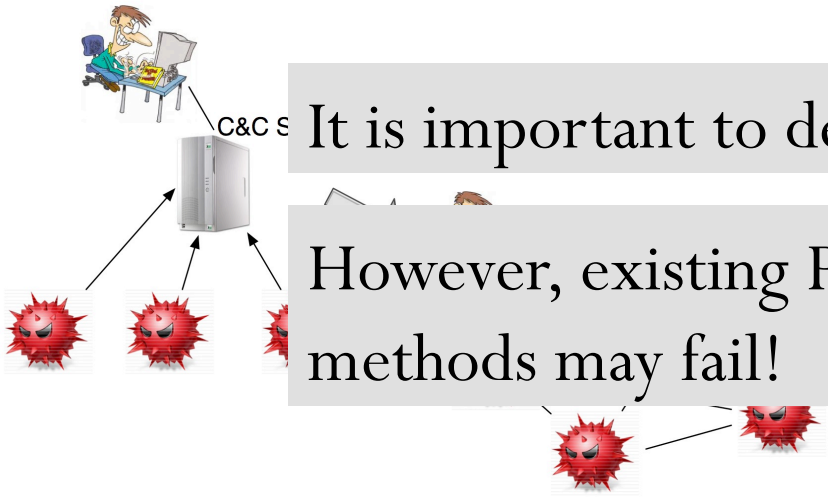
- Botnet
  - A collection of bots controlled by a botmaster via a ***command and control (C&C)*** channel
  - Botnets serve as the infrastructures for a variety of attacks
    - Spam
    - Identity Theft
    - DDoS
    - …



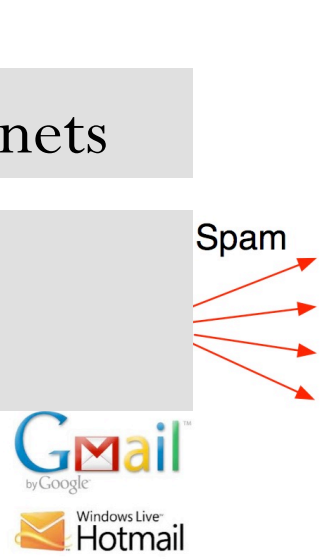
Introduction

- Botnets Evolve

- C&Cs
  - Centralized -> P2P



- Attacks
  - Noisy -> Stealthy

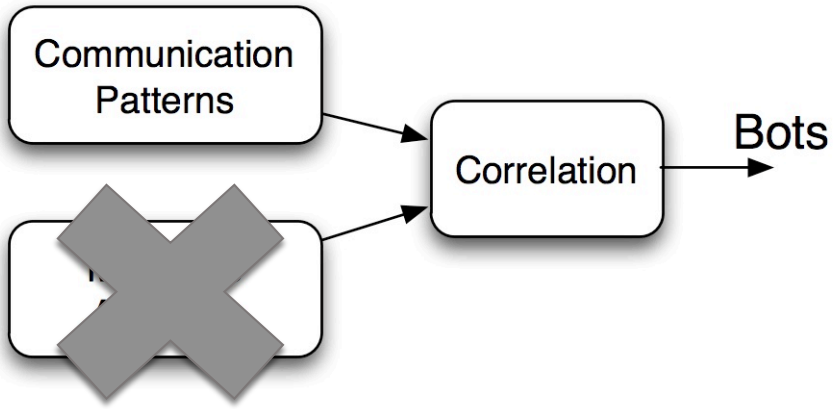


It is important to detect stealthy P2P botnets

However, existing P2P botnet detection methods may fail!

Introduction

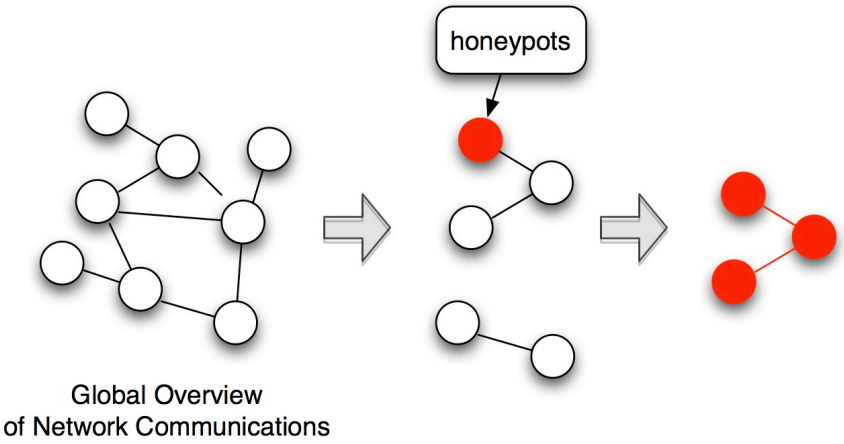
- Existing Method 1
- BotMiner [Gu et al Security 08]
  - Correlates similar activities in communication plane and attack plane



- However, stealthy attacks may not be observable, making the correlation ineffective!

Introduction

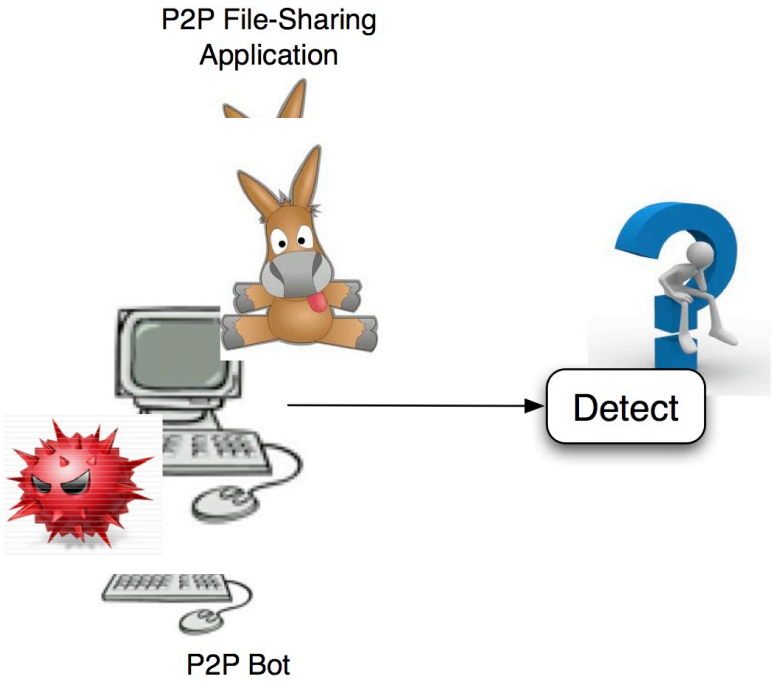
- Existing Method 2
- BotGrep [Nagaraja et al Security 10]
  - Uses graph analysis to identify P2P networks
  - Requires global overview of network communications
  - Requires bootstrap information (e.g., honeypots)



- However, it is challenging to get
  - Network global overview
  - A priori detection results
- Resulting in limited usage in practice!

Introduction

- Existing Method 3
- Telling P2P File-Sharing and Bots Apart [Yen et al ICDCS 10]
  - Differentiate P2P bots from P2P file-sharing clients

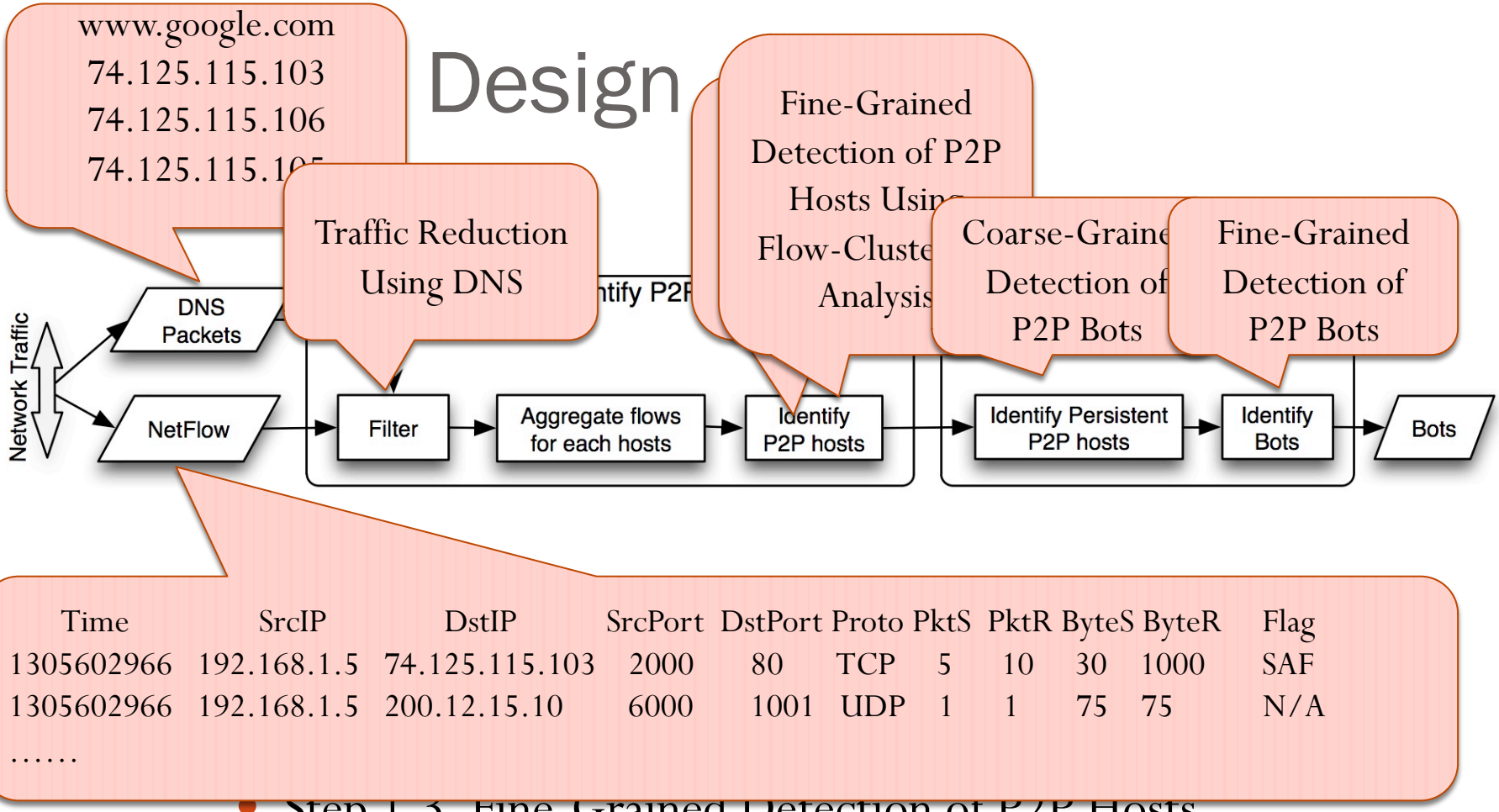


- However, the method may fail if bot-compromised host is running a P2P file-sharing application

Introduction

- We need a new method to detect stealthy P2P botnets!
- Design Targets:
  - Detect P2P bots without observing attack activities
  - Detect P2P bots even if the underlying hosts run legitimate P2P applications

Design



- Step 1.3. Fine-Grained Detection of P2P Hosts
- Phase II: Identify P2P Bots
  - Step 2.1. Coarse-Grained Detection of P2P Bots
  - Step 2.2 Fine-Grained Detection of P2P Bots

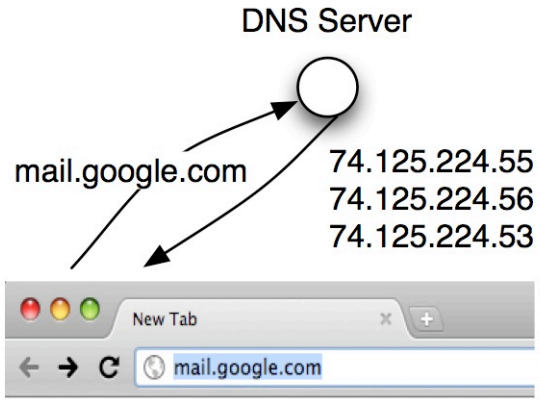
System Design

- Use real-world P2P traces to motivate the system design and parameter selection
  - 5 popular P2P applications
  - 24 hours

P2P Applications	Version	Protocol
Bittorrent	6.4	Bittorrent
Emule	0.49c	Kademlia
Limewire	5.4.8	Gnutella&Bittorrent
Skype	4.2	Skype
Ares	2.1.5	Gnutella&Bittorrent

System Design – Phase I: Identify P2P Hosts

- Step 1.1 Traffic Reduction
  - Observation: The peer IP addresses of P2P connections are obtained from the overlay routing tables instead of DNS responses.
  - System Design: Filter out the flows, whose destination IP addresses are resolved from DNS queries.



Trace	Percentage of flows associated with no-DNS DstIPs
T-Bittorrent	96.85%
T-Emule	99.99%
T-Limewire	99.97%
T-Skype	99.93%
T-Ares	99.99%

System Design – Phase I: Identify P2P Hosts

- Step 1.2 Coarse-Grained Detection of P2P Hosts
  - Observation: The significant node churn of P2P networks introduces a large number of failed connections.
  - System Design: Identify hosts that initiate more than θ_o failed connections for each day.

Trace	# of Failed Conns Per hour
T-Bittorrent	1602
T-Emule	318
T-Limewire	1278
T-Skype	81
T-Ares	489

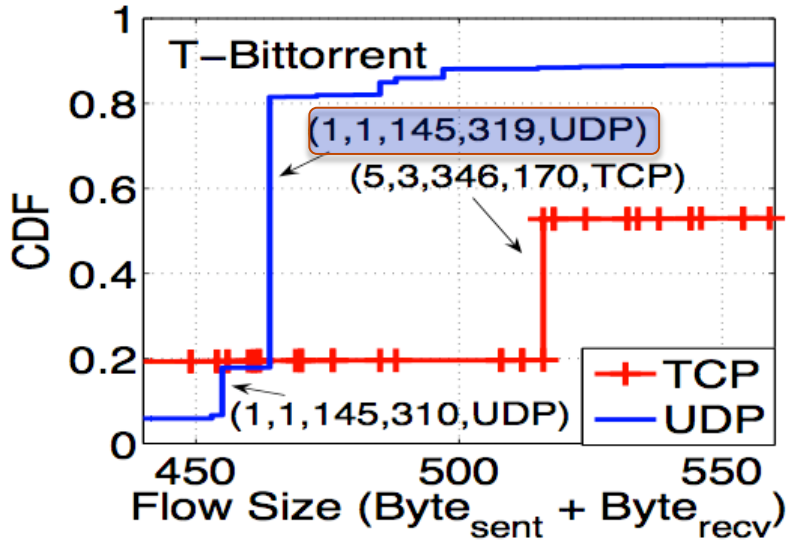
Conservatively set $\theta_o = 10$

System Design – Phase I: Identify P2P Hosts

- Step 1.3 Fine-Grained Detection of P2P Hosts
  - Observation:
    - A P2P application automatically generates a large number of control flows (e.g., ping/pong) ***to a large number of peers***, as long as it is online.
    - The control flows used for the same purpose (e.g., ping/pong or peer discovery) ***share similar size***.
  - System Design:
    - Aggregate flows with similar sizes to clusters respectively.
    - Define a cluster as ***a fingerprint cluster***, if the number of unique BGP prefixes of the destination IP addresses for its flows is greater than θ_{BGP} (=50).
    - Claim a host as P2P host if it has at least one fingerprint cluster.

System Design – Phase I: Identify P2P Hosts

- An example of control flows

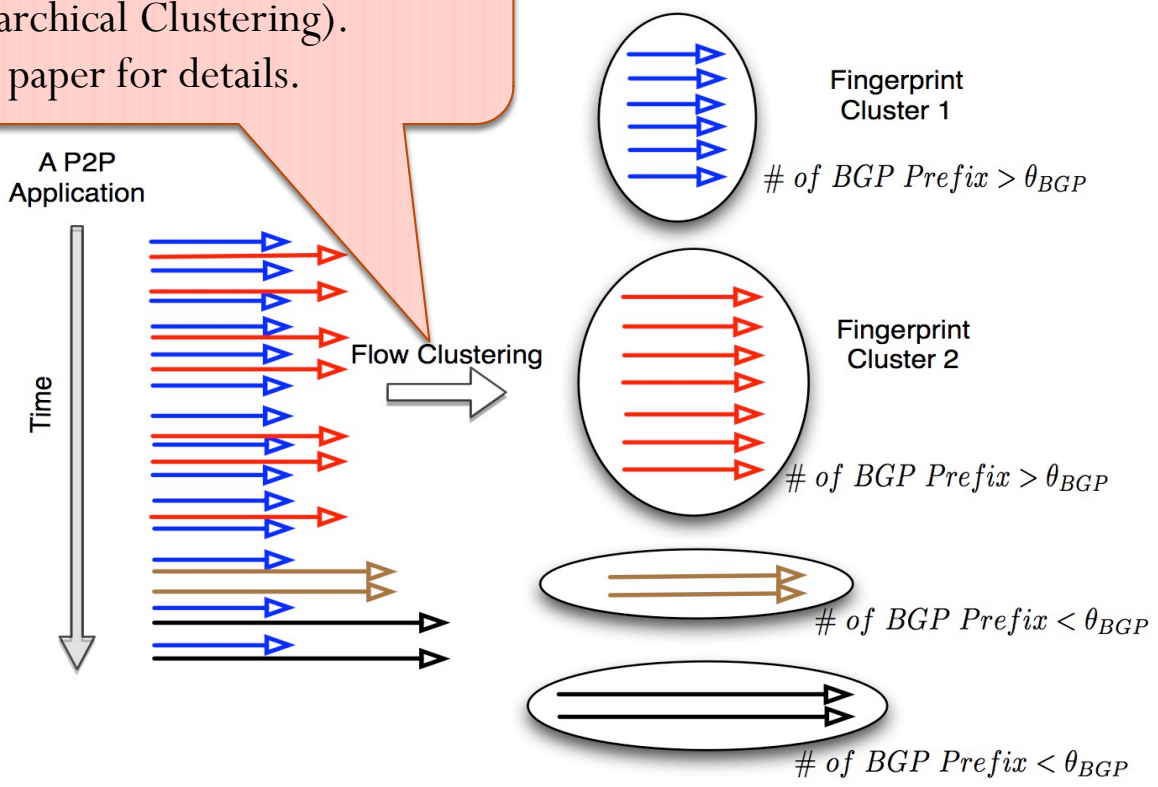


Fingerprints	flows	outgoing content	incoming content	description
1 1 145 319, UDP	1	d1:ad2:id20:. . . find_node1:. . . :y1:qe	d1:rd2:id20:. . . nodes208:. . . :y1:re	peer discovery
	2	d1:ad2:id20:. . . find_node1:. . . :y1:qe	d1:rd2:id20:. . . nodes208:. . . :y1:re	
	. . .	d1:ad2:id20:. . . find_node1:. . . :y1:qe	d1:rd2:id20:. . . nodes208:. . . :y1:re	

System Design – Phase I: Identify P2P Hosts

- Step 1.3. Flow-Clustering analysis for identifying fingerprint

•Scalable clustering scheme (Birch +Hierarchical Clustering).
•Refer paper for details.



System Design – Phase II: Identify P2P Bots

- Step 2.1 Coarse-Grained Detection of P2P Bots
  - Motivation:
    - A bot usually is active as long as its underlying system is power on
    - A legitimate P2P application may only be used for a short period of time
  - System Design:
    - Identify P2P clients that are ***persistently active*** compared to its underlying host

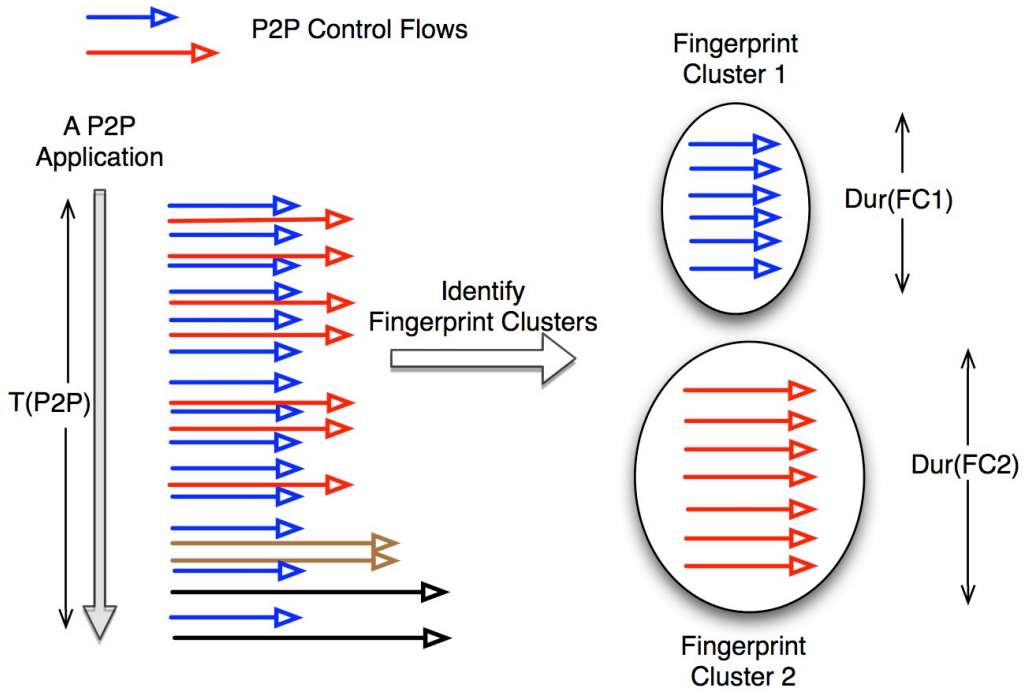
$$T(P2P) / T(Host) > 0.5$$

- Challenge: How can we accurately estimate the active time of a P2P application (i.e., T(P2P))?
- Solution: Leverage obtained fingerprint clusters!

System Design – Phase II: Identify P2P Bots

- Estimate the active time of a P2P application (T(P2P))

$$Dur(P2P) = MAX(Dur(FC_1), Dur(FC_2)\ldots\ldots Dur(FC_n))$$



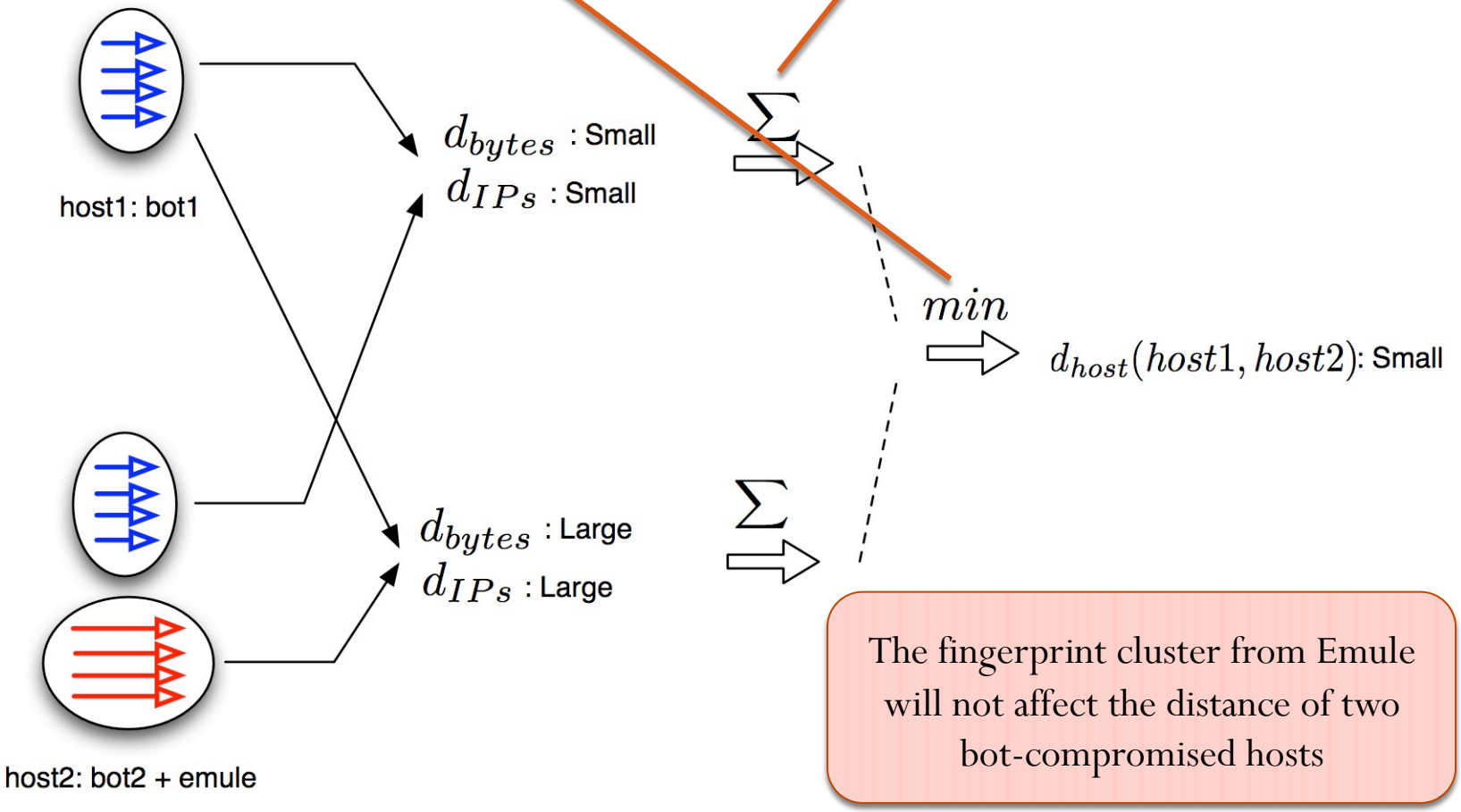
Trace	Actual P2P Active Time (T(P2P))	Estimated P2P Active Time (Dur(P2P))
T-Bittorrent	24hr	24hr
T-Emule	24hr	24hr
T-Limewire	24hr	24hr
T-Skype	24hr	24hr
T-Ares	24hr	24hr

- T(P2P) ≈ Dur(P2P).
- Dur(P2P) can accurately approximates T(P2P)!

System Design – Phase II: Identify P2P Bots

- Fine-Grained Detection of P2P Bots
  - Motivation
    - Same P2P applications, such as P2P bots of the same botnet, use the same P2P protocol, resulting in similar/same fingerprint clusters
    - P2P bots in the same botnet search for the same commands/contents. Therefore the search paths from bots tend to converge, resulting in a large overlap of peers.
    - Legitimate P2P clients are driven by different users, searching for different contents. Therefore the search paths for legitimate clients tend to diverge, resulting in a small overlap of peers.

$$dist(h_a, h_b) = \min_{i,j}(\lambda * \frac{d_{bytes}(FC_i^{(a)}, FC_j^{(b)}) - \min_B}{\max_B - \min_B} + (1-\lambda) * d_{IPs}(FC_i^{(a)}, FC_j^{(b)}))$$

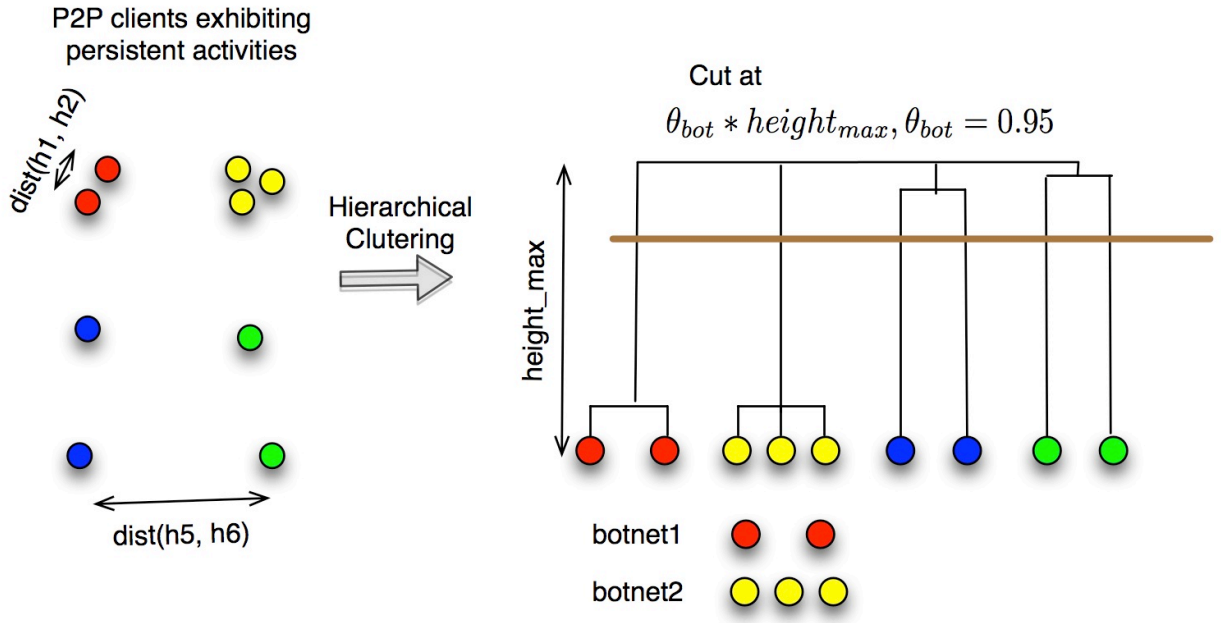


The fingerprint cluster from Emule will not affect the distance of two bot-compromised hosts

bot-compromised system runs an Emule application

System Design – Phase II: Identify P2P Bots

- Fine-Grained Detection of P2P Bots
  - System Design: If two persistently active P2P clients have small distance, we identify them as bots



System Evaluation

- Experimental Setup

Data Sources	Description	P2P Clients/Traces
College networks	Passively collected for 24 hours; Netflows with 200 bytes payload between internal and external networks; DNS responses with IP addresses	3 Bittorrents (ground truth by signatures); 5 Skypes (potential)
5 P2P applications	Run each application ***simultaneously*** on two VMs for 24 (or 5) hours	10 traces totally
2 P2P botnets	Execute the malware binaries of Storm/Waledac for 24 hours	13 Storm bots; 3 Waledac bots

System Evaluation

- Experimental Design
  - Identifying and Profiling P2P applications
  - Detecting P2P bots
    - With being mixed with legitimate P2P data
    - Other situations are discussed in the paper including
      - Without being mixed with legitimate P2P data
      - Only two bots from each botnet
      - No bot (clean network)

System Evaluation

- Identifying and Profiling P2P Applications
  - DNS-based filter eliminates 2/3 hosts in the college networks!
  - Effectively identify and profile all the P2P applications!
    - 3 Bittorrent hosts in college networks
    - 5 potential Skypes in college networks
    - 10 P2P application instances we have run
    - 3 Waledac P2P bots
    - 13 Storm bots

Let us have a close look at Fingerprint Clusters!

System Evaluation

- Identifying and Profiling P2P Applications
  - Fingerprint cluster summaries for Bittorrent clients in college networks
  - Fingerprint cluster summary: average number of pkt_sent/recv, byte_sent/recv, and protocol for one fingerprint cluster

Fingerprints
Ground truth

	$(\overline{Pkt_s}, \overline{Pkt_r}, \overline{Byte_s}, \overline{Byte_r}$, proto)
T-Bittorrent	1 1 145 319, UDP 1 1 109 100, UDP 1 1 146 340, UDP 5 3 346 170, TCP 1 1 145 310, UDP
T-Bittorrent-2	1 1 145.01 317.66, UDP 1 1 109 100, UDP 1 1 146 342, UDP 5 3 346 170, TCP 2 2 466 461, UDP

Fingerprints identified in
the college network

	$(\overline{Pkt_s}, \overline{Pkt_r}, \overline{Byte_s}, \overline{Byte_r}$, proto)
BT1@C	1 1 109 100, UDP 1 1 109 91, UDP 1 1 104 178, UDP 1 1 319 145, UDP 1 1 145 319, UDP
BT2@C	1 1 145 319, UDP 1 1 75 75, UDP 1 1 65 65, UDP
BT3@C	7 6 1118 1767, TCP

System Evaluation

- Identifying and Profiling P2P Applications
  - Fingerprint cluster summaries for Skype clients in college networks

Fingerprints
Ground truth

$(\overline{Pkt_s}, \overline{Pkt_r}, \overline{Byte_s}, \overline{Byte_r}, \text{proto})$

Skype	1 1 74.58 60, UDP 1 1 78 60, UDP 1 1 75 60, UDP 1 1 76 60, UDP 1 1 79 60, UDP

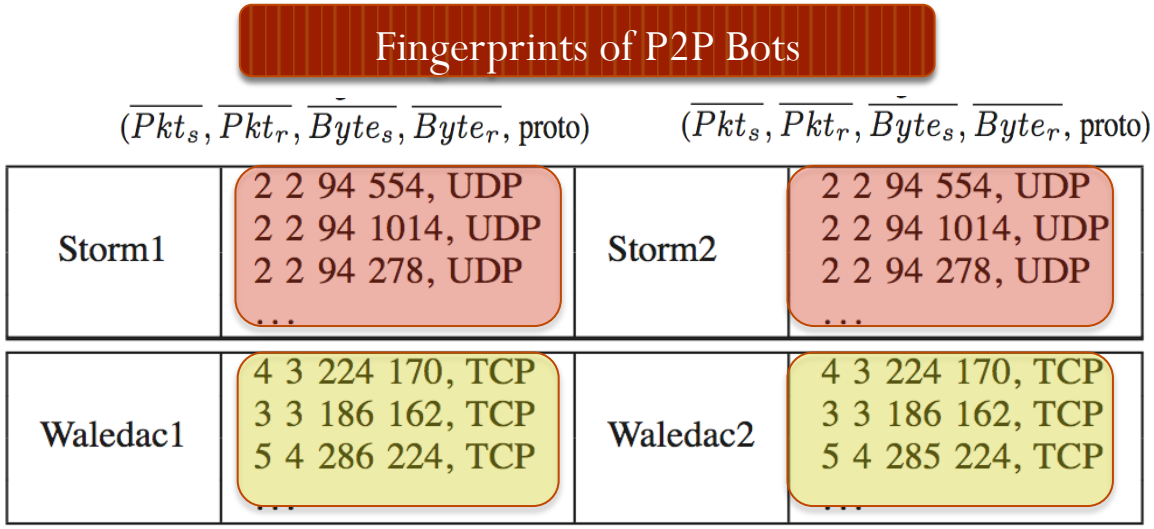
Fingerprints identified in the college network

$(\overline{Pkt_s}, \overline{Pkt_r}, \overline{Byte_s}, \overline{Byte_r}, \text{proto})$

Skype1@C	1 1 73 60, UDP 1 1 76 60, UDP 1 1 75 60, UDP 1 1 72 60, UDP 1 1 74 60, UDP
Skype2@C	1 1 75 60, UDP 1 1 74 60, UDP 1 1 76 60, UDP
Skype3@C	1 1 72 60, UDP 1 1 74 60, UDP 1 1 79 60, UDP 1 1 76 60, UDP
Skype4@C	1 1 73 60, UDP
Skype5@C	1 1 74 60, UDP 1 1 75 60, UDP

System Evaluation

- Identifying and Profiling P2P Applications
  - Fingerprint cluster summaries for P2P bots



The fingerprint clusters can effectively identify and profile different P2P applications!

System Evaluation

- Detecting P2P bots
  - Overlaying legitimate P2P data and P2P botnet data to college network data.
  - half of bot-compromised hosts run legitimate P2P applications.

Bot	P2P App	Before Overlay (Bot)			After Overlay (Bot+P2PApp)		
		# of flows	# of DstIPs	avg flow size	# of flows	# of DstIPs	avg flow size
Waledac1	Emule1	341784	850	12829	452645	15338	55688
Waledac2	BT2@C	319119	760	11372	361135	1359	348708
Storm1	Limewire1	200237	6390	1342	429458	16635	1714
Storm2	BT3@C	275451	7319	1337	310667	8307	3381
Storm3	Bittorrent2	133955	5584	1344	432464	23261	172945
Storm4	Skype4@C	171471	7277	1280	199101	7520	1266
Storm5	Skype1	164917	6686	1328	214548	13137	1307
Storm6	Ares1	220459	6618	1307	238063	8543	6244

TABLE IX: Bot Traces Overlapped with P2P Application Traces

Bot traffic profiles are significantly skewed by the legitimate P2P data, which could make the existing detection approach such as [Yen etal ICDCS10] ineffective.

System Evaluation

- Detecting P2P Bots
  - Fingerprint clusters of P2P bots overlapped with legitimate P2P applications

Fingerprints of P2P Bots Mixed with P2P Apps

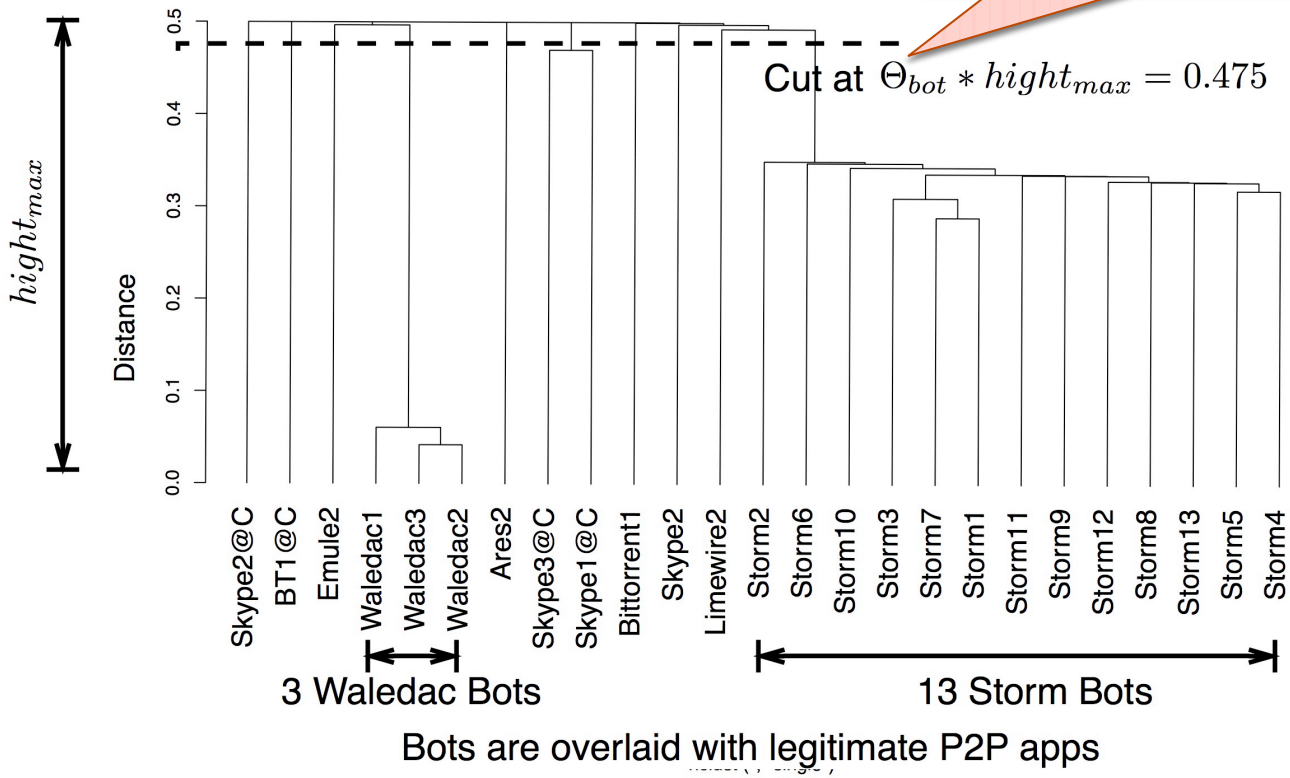
	$(\overline{Pkt_s}, \overline{Pkt_r}, \overline{Byte_s}, \overline{Byte_r}, \text{proto})$
Waledac2+BT2@C	1 1 145 319, UDP (Bittorrent) 4 3 224 170, TCP (Waledac) 3 3 185 162, TCP (Waledac) 1 1 75 75, UDP (Bittorrent) ...
Storm4+Skype4@C	2 2 94 554, UDP (Storm) 2 2 94 1014, UDP (Storm) 1 1 73 60, UDP (Skype) ...

Fingerprint clusters can effectively profile P2P applications, even if the underlying host is running different applications.

System Evaluation

- Detecting P2P Bots

Refer paper for more evaluation of θ_{bot}



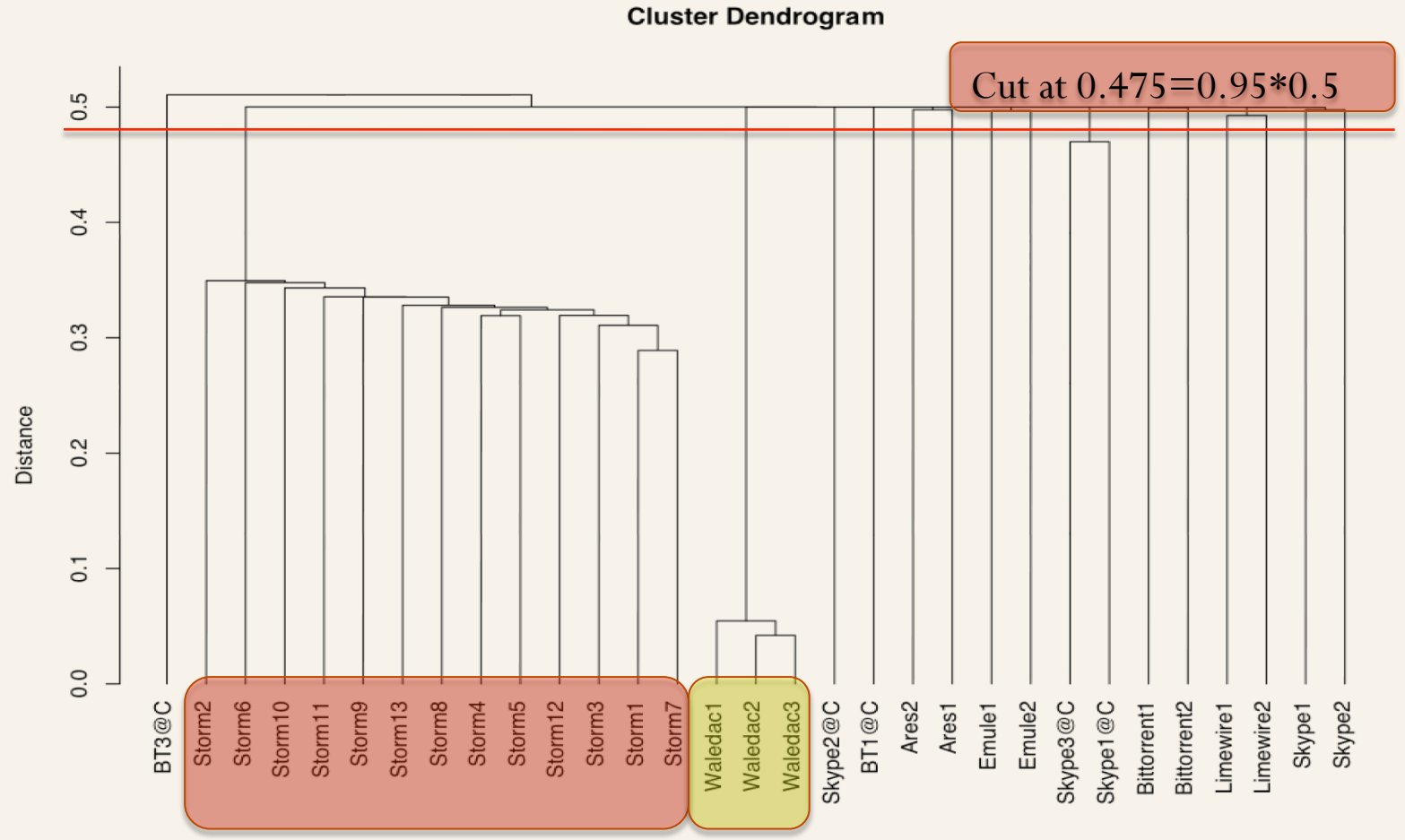
Detection Rate	100%
False Positive Rate	0.2% (2/953)

Conclusion

- A new P2P traffic detection algorithm
  - identify and *profile* P2P applications
- A novel P2P botnet detection system
  - Detect P2P bots without observing malicious activities
  - Detect P2P bots even if legitimate P2P applications are running on the underlying systems
  - High detection rate and low false positive rate

End

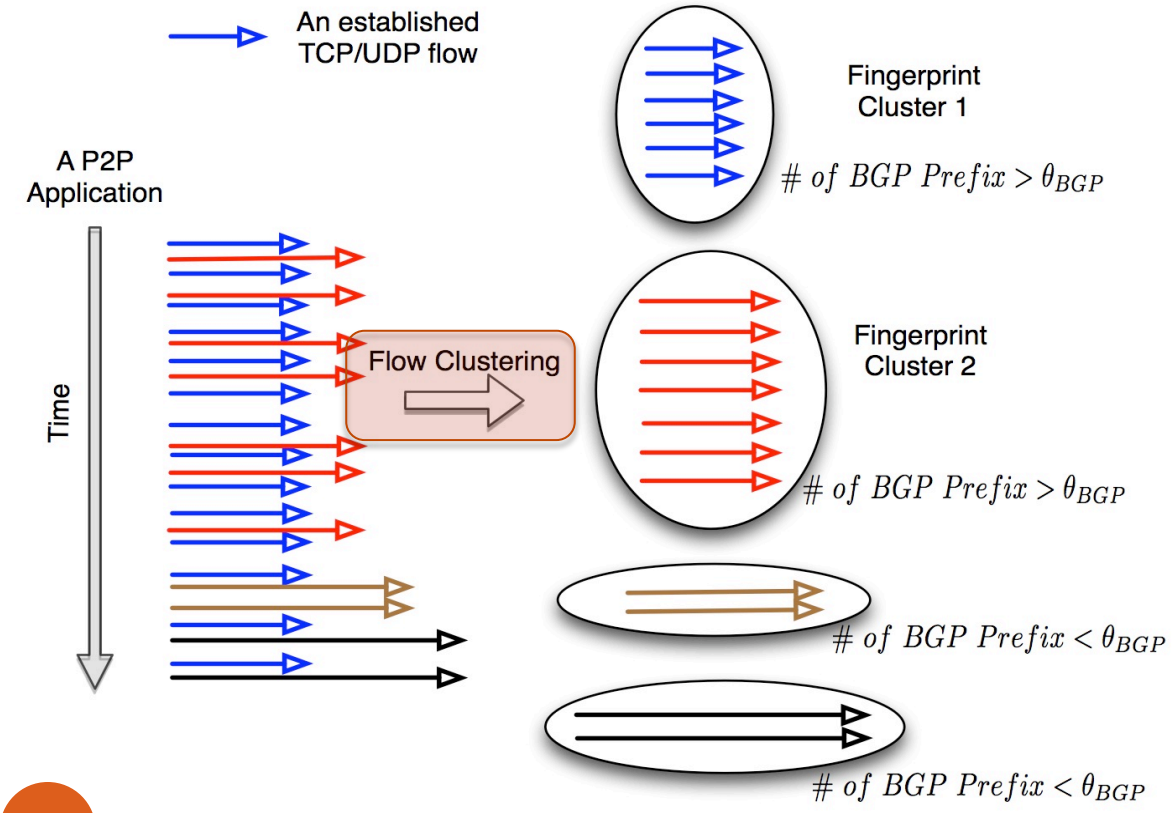
- Thanks!

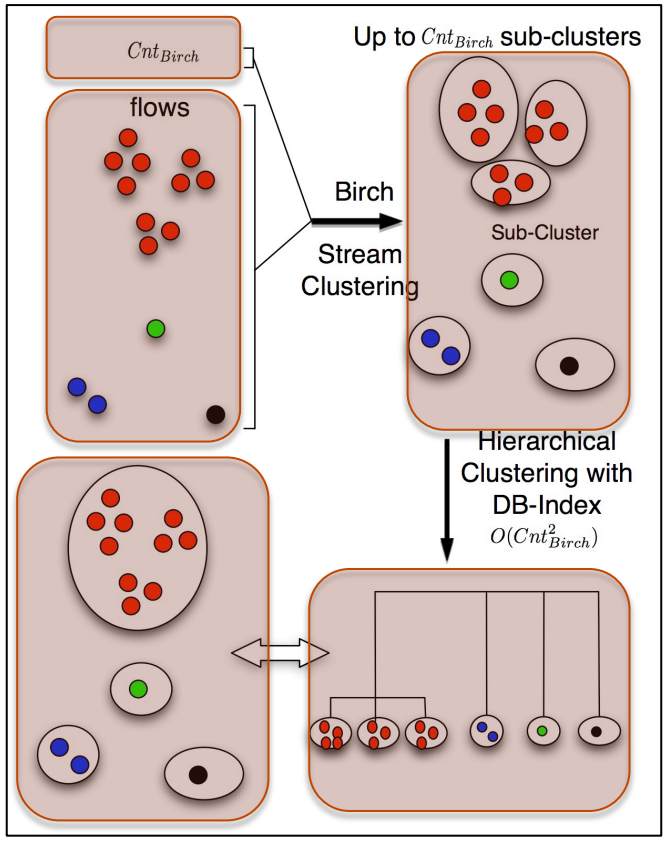


Detection Rate	**100%**
False Positive Rate	0.2% (2/953)

System Design – Phase I: Identify P2P Hosts

- Two steps of clustering analysis to generate fingerprint clusters.

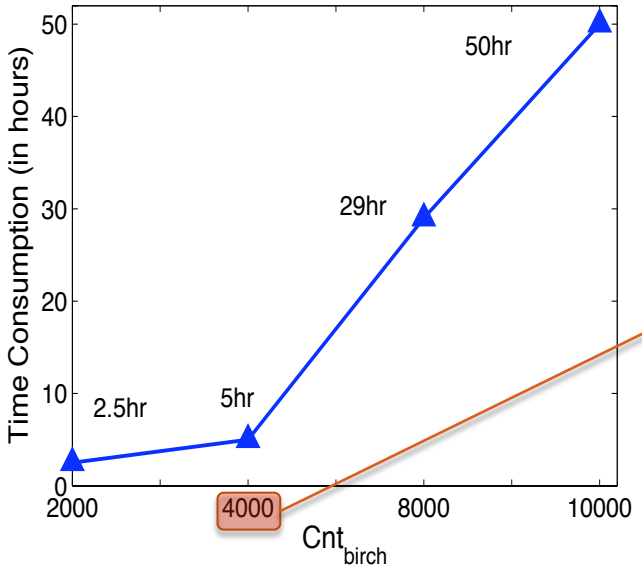




$Cnt_{Birch} = 4000$

System Evaluation

- Experimental Results
  - Scalability Regarding Time Consumption



Our Experimental Configuration

		Θ_{bot}						
Cnt_{birch}	-	0.1	0.3	0.5	0.7	0.8	0.9	0.95
2000	DR FP	0 0	0 0	2/16 0	3/16 0	16/16 0	16/16 0	16/16 2/953
4000	DR FP	2/16 0	3/16 0	3/16 0	16/16 0	16/16 0	16/16 0	16/16 2/953
8000	DR FP	2/16 0	3/16 0	3/16 0	16/16 0	16/16 0	16/16 0	16/16 2/953
10000	DR FP	2/16 0	3/16 0	3/16 0	16/16 0	16/16 0	16/16 0	16/16 2/953

A small Cnt_{birch} increases system performance, but may decrease accuracy of fingerprint clusters.

Achieving good detection rates for a large range of Θ_{bot} (e.g., $[0.7 - 0.95]$).
Achieving good scalability by tuning Cnt_{birch} .