

Unsupervised host behavior classification from connection patterns

Pierre BORGNAT

CNRS – ENS Lyon, Laboratoire de Physique (UMR 5672)

LIP6 – 03/2010



gipsa-lab



Internet Initiative Japan



Joint-work with...

- Guillaume DEWAELE, Patrice ABRY
(Sisyph, Physics lab., ENS Lyon)
- Olivier MICHEL (GIPSA-lab, INPG, Grenoble)
- Kensuke FUKUDA, Romain FONTUGNE (NII, Tokyo)
- Yosuke HIMURA (University of Tokyo)
- Kenjiro CHO (IIJ, Tokyo)

Objectives

[Context: Passive Monitoring of TCP/IP traffic on a link]

Long-term motivations

- Internet is a living beast → no accurate models
 - Diversity of expected traffic: http, P2P, mail, DNS,...
 - Variety of conditions: used bandwidth, congestion,...
 - Frequent anomalies: scans, viruses&worms, DDoS,...
- Provide tools & information to network administrators:
 - Methods of anomaly detection
 - Robust Longitudinal analyses over spans of years
 - Tools of traffic classification
 - ...
- MAWI dataset: more than 9 years of daily traces in pcap
- Statistical approaches to these topics

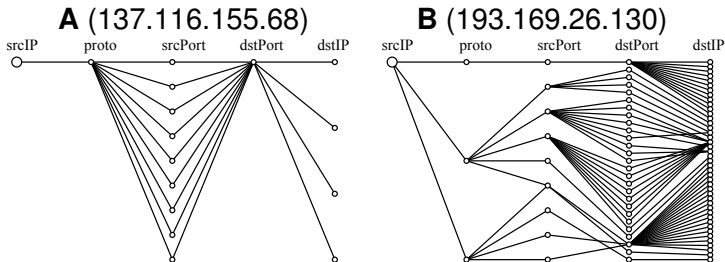
Requirements for “traffic classification”

- High-speed links of Backbones:
 - No bi-directionality
 - No packet payload (useful for a posteriori & online work)
 - Robustness to sampling
- Unsupervised classification:
 - Allow finding new classes of traffic
 - No need for labelled training set
- Host-level analysis
 - vs. usually: flow or packet-level approaches
 - Strengths: cases of mix traffic; network administrator point of view (→ IP)

Other existing approaches

- Rule-based methods:
 - Should be pre-defined
 - Not adaptive to novelties
 - An interesting state-of-the-art method:
“BLINC: Multilevel Traffic Classification in the Dark”,
Karagiannis *et al.*, SIGCOMM 2005.
- Statistics-based methods:
 - Many supervised methods (limits: need for a training set)
 - Some unsupervised methods: none at the host-level

Inspiration: Host connection described with Graphlets



However, some drawbacks:

- Representation in infinite-dimension space
- Hosts with mixed types of traffic → complex graphlets

Set of quantitative features of connection patterns

- ***I. Network connectivity***
 - i) the number of peers (or destination IPs)
 - ii) the number source ports, divided by the # of peers (dst IPs)
 - iii) the number of destination ports, divided by the # of peers (dst IPs)
- ***II. Connection dispersion in the network.***
 - iv) the ratio of the entropies of the second and fourth bytes of IPdst
 - v) the ratio of the entropies of the third and fourth bytes
- ***III. Host traffic content.***
 - vi) the mean number of packets per flow
 - vii) the percentage of small size packets (≤ 144 bytes)
 - viii) the percentage of large size packets (≥ 1392 bytes)
 - ix) the entropy of the distribution of medium size packets

These features obey a Parsimony / Relevance trade-off.

Comments on some feature

- ***I. Network connectivity***

- i) the number of peers (or destination IPs)
 - ii) the number source ports, divided by the # of peers (dst IPs)
 - iii) the number of destination ports, divided by the # of peers (dst IPs)
- ⇒ Classical features: popularity of the host (client, server, host in overlay network,...)

- ***II. Connection dispersion in the network.***

- iv) the ratio of the entropies of the second and fourth bytes of IPdst
 - v) the ratio of the entropies of the third and fourth bytes
- ⇒ For the functional/social role of the host: quantification of connection dispersion, or the spreading in the IP space of the peers
- Entropy $S = -\sum_i p_i \log p_i$ ← Spiky vs. Flat distribution p_i

Comments on some feature

- ***I. Network connectivity***

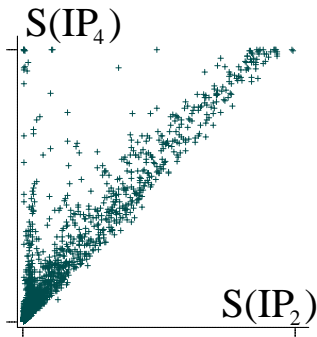
- i) the number of peers (or destination IPs)
 - ii) the number source ports, divided by the # of peers (dst IPs)
 - iii) the number of destination ports, divided by the # of peers (dst IPs)
- ⇒ Classical features: popularity of the host (client, server, host in overlay network,...)

- ***II. Connection dispersion in the network.***

- iv) the ratio of the entropies of the second and fourth bytes of IPdst
 - v) the ratio of the entropies of the third and fourth bytes
- ⇒ For the functional/social role of the host: quantification of connection dispersion, or the spreading in the IP space of the peers
- Entropy $S = - \sum_i p_i \log p_i$ ← Spiky vs. Flat distribution p_i

Connection dispersion: entropy of IP bytes

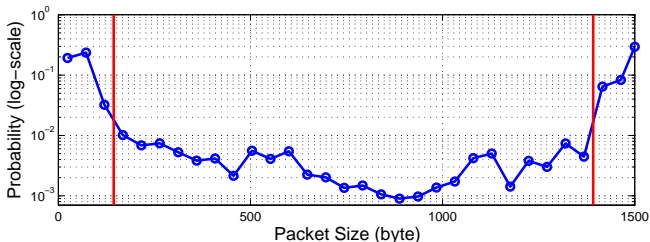
- IPv4: distribution of each bytes inherited from the network



- Each dot = a host. Two areas:
 - $S(IP_2) \ll S(IP_4)$
 - $S(IP_2) \lesssim S(IP_4)$
- To distinguish them: $F_{iv} = S(IP_2)/S(IP_4)$

Features for Host traffic content

- “Traffic content” yet without any payload information
- **III. Host traffic content.**
 - **vi)** the mean number of packets per flow ← classical
 - **vii)** the percentage of small size packets (≤ 144 bytes)
 - **viii)** the percentage of large size packets (≥ 1392 bytes)
 - **ix)** the entropy of the distribution of medium size packets
- \Rightarrow Histogram of packet-sizes (steps of 38 bytes)



- Small and large packets: They respectively account for 46% and 44% of the packets.

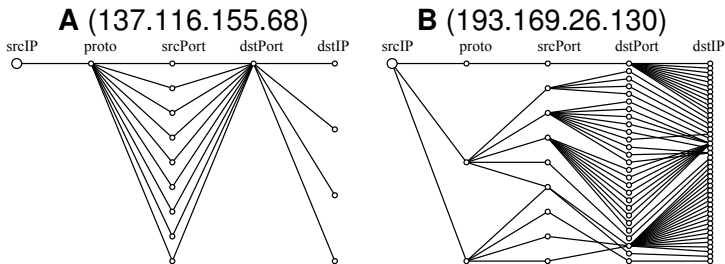
Normalization of the feature space

- To balance the relative importance of the features, non-linear renormalization:

$$f_n = (2/\pi) \arctan(F_n/R_n)$$

- References R_n :
 - $R_i = 10$, corresponding to average number of peers
 - $R_{ii} = R_{iii} = F_i$, the actual number of peers
 - $R_{vi} = 100$, average size of the flows
 - F_{ix} (entropy of midsize pkts) is scaled into $[0, 1]$ with a division by $\log K$ (where K is the number of bins used in the estimation in the packet size distribution).
 - F_{iv} and F_v already normalized (ratios), as well as F_{vii} and F_{viii} (percentage of traffic)

Comparison: Graphlet vs. Feature space



Host	Peers	Src ports	Dst ports	S(2/4)	S(3/4)	Flow len.	Small	Large	S(mid)
A	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐
B	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐
B (rcv)	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐	┌──┐

- Host **A** is mostly doing HTTP requests over TCP: small #1 of peers and src ports, targets a single same dst port (port 80), dominant proportion of small packets
- Host **B**, hard time to interpret the graphlet !...
From the features: mix type of traffic (P2P traffic with many peers with the emission of a Ping-flood aiming at a large # of destination hosts)

Choice of a classification procedure

- Requirements:
Unsupervised method, arbitrary # of classes, works in high-dimensional space (at least 9D), non-convex clusters
- Methods based on Minimum Spanning Tree work well:
 - MST for a given set of nodes = the fully connected acyclic graph whose edge total length is minimal
 - Clustering from cutting edges → yields a posteriori and data-driven clusters
 - Theoretical attractive features:
 - MST seen as entropy estimators;
 - relationship between MST features and manifold properties (with high dimensional data lying on a lower dimensional manifold, MST can estimate it)
 - MST constructions easily implemented, with a computational burden maintained in $O(N \log N)$ operations

Choice of a classification procedure

- Requirements:
Unsupervised method, arbitrary # of classes, works in high-dimensional space (at least 9D), non-convex clusters
- Methods based on Minimum Spanning Tree work well:
 - MST for a given set of nodes = the fully connected acyclic graph whose edge total length is minimal
 - Clustering from cutting edges → yields a posteriori and data-driven clusters
 - Theoretical attractive features:
 - MST seen as entropy estimators;
 - relationship between MST features and manifold properties (with high dimensional data lying on a lower dimensional manifold, MST can estimate it)
 - MST constructions easily implemented, with a computational burden maintained in $O(N \log N)$ operations

Steps of the clustering procedure: edge-cutting in MST

I) Compute MST, using a greedy algorithm

Note: some features are integers, the feature space is made continuous by adding random values uniformly spread in $[0, 1]$

II) First Clustering: cut all lengths larger than $T = 0.25$

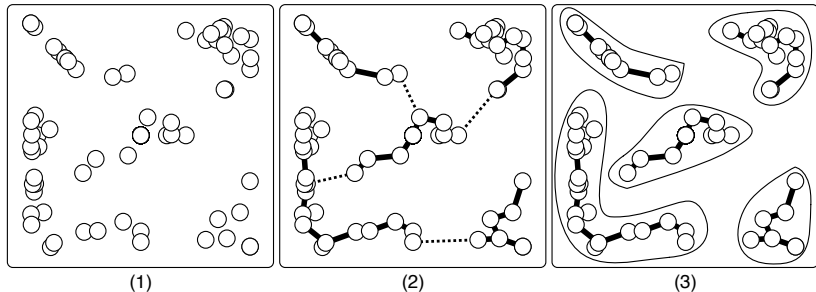
Note: Clustering by cutting edges in MST is known to be unstable in the presence of outliers \Rightarrow Third step

III) Identify dense clusters:

- Core of cluster = subtrees of cardinality ≥ 10 , whose nodes are connected by edges of length $\leq T' = 0.05$
- Clusters are obtained by growing from cores, in the cut MST

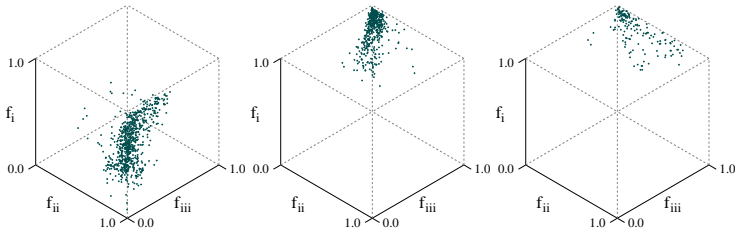
Steps of the clustering procedure: edge-cut with MST

- (1) A set of hosts into a (reduced 2D) feature space



- (2) the MST with the longest edges in dashed lines
- (3) Step //: edge cutting procedure, yields the clusters

Some clusters (in projected space)



- Some clusters (C1, S1 and S3) displayed in projected space on features i , ii and iii (only)
- Each dot is one host of the given cluster
- Note the non-convex and intricate shapes of the clusters

Presentation of the results

- MAWI Dataset: 15-min pcap traces (anonymized, no payload)
- On a 1 Gbps transpacific link (Japan – US)
- Clustering on 7 days in Jan. 2008 ⇒ Unsupervised clusters
- Classification on those for 50 other traces (5 per month) in 2008

- Kept: results for hosts that send at least 1500 packets

- Validation: Ground truth? None... Payload? None...
 - Manual inspection
 - Cross-validation: port-based classifier, BLINC

Features of the identified clusters

Id	Peers	Src ports	Dst ports	S(2/4)	S(3/4)	Flow lenght	Small	Large	S(mid)	#Hosts
T_1										11637
T_2										6344
T_3										1626
T_4										1591
T_5										572
T_6										986
T_7										586
C_1										7875
C_2										2765
C_3										524
C_4										2389
C_5										1566
C_6										608
C_7										530
S_1										5383
S_2										1772
S_3										1760
S_4										991
S_5										690
S_6										4225
S_7										4056
S_8										1694
S_9										476
S_{10}										442
P_1										4461
P_2										560

Analysis of identified clusters

- Clusters labelled manually from the constated hosts' behaviors
- **T** (Transfer): with traffic using few ports on both sides
 - $T_1 - T_5$: 1-to-1 connections, HTTP and P2P, long flows
 - $T_6 - T_7$: many peers, Ping, DNS, P2P (fixed ports) and many anomalies
- **C** (Clients): from many ports to a limited numbers of dst ports
 - C_1 to C_3 : clients that connect to many different servers
 - C_4 to C_7 : connection to less servers
- **S** (Servers): large # of ports, from a limited # of ports (e.g., web servers)
 - S_1 to S_5 popular servers; S_6 to S_{10} small # of peers
- **P** (P2P): large # of both dst and src ports, hidden P2P
 - P_1 : typical leechers; P_2 mix of activities (e.g., host **B**)

Cross-validation with port-based analysis

- Independently, a port-based classifier is applied
→ known to be OK for legacy applications
If multiple traffic: use the dominant one
- Addition 1: ratio of SYN-flag to detect SYN-flooding
(a most frequent event in the traces)
- Addition 2: Mix traffic if the dominant port accounts for less than 50% of traffic, or more than one type account for each more than 20% of traffic
- Addition 3:
Anomaly detection (only if needed for interpretation)

Cross-validation with port-based analysis

Id	HTTPr	HHTPa	P2P	Ping	SYN	SMTPr	SMTPa	DNSr	DNSa	SSHr	SSHa	Mix	#Hosts
T_1	6771	121	3357	427	1	3	59	55	53	46	24	41	11637
T_2	3	5581	364	0	0	112	0	0	0	0	8	5	6344
T_3	16	539	802	9	0	7	0	0	0	3	4	14	1626
T_4	2	197	892	250	0	6	0	0	43	2	16	16	1591
T_5	7	22	382	13	0	6	0	0	0	2	8	15	572
T_6	51	21	41	622	0	0	16	133	58	2	1	7	986
T_7	0	0	583	1	0	0	0	0	0	0	0	0	586
C_1	6138	0	130	3	18	115	0	119	0	43	2	1003	7875
C_2	2271	2	215	16	0	1	1	37	0	12	0	57	2765
C_3	69	0	0	78	220	11	0	83	0	0	0	25	524
C_4	2057	4	144	1	3	18	0	5	0	1	2	49	2389
C_5	751	0	248	0	3	49	0	1	0	17	0	151	1566
C_6	147	0	60	0	10	0	0	1	0	1	0	309	608
C_7	224	0	30	0	8	2	0	0	0	3	0	193	530
S_1	0	4648	171	0	0	1	0	0	16	0	2	340	5383
S_2	0	1637	65	0	0	2	0	0	0	0	3	22	1772
S_3	12	369	257	11	0	0	442	212	29	1	60	337	1760
S_4	14	221	193	6	1	0	309	14	124	0	26	47	991
S_5	7	561	47	0	0	10	0	0	0	1	2	19	690
S_6	0	3849	45	0	0	1	0	0	3	0	2	123	4225
S_7	17	3578	191	0	0	63	0	0	0	0	4	32	4056
S_8	0	302	33	0	0	0	116	0	37	0	1136	17	1694
S_9	0	455	7	0	0	0	0	0	0	0	0	3	476
S_{10}	0	421	11	0	0	0	0	0	0	0	0	3	442
P_1	719	186	523	12	44	111	272	239	38	0	29	1922	4461
P_2	9	5	235	0	15	5	0	1	0	0	5	251	560

Comments: Cross-validation with a “Ports”

- The table is relatively sparse: good coherence
- Identified clusters: they fall mostly in the proper “port-based” class
 - T_1 = requests in HTTP and P2P; T_2 = answers over HTTP; T_3 and T_4 = P2P plus some web browsing,
 - C and S well separated in requests / answers
 - P = P2P + mix, not easily in a “port-based” class
- Clusters with a large # of anomalies (T_4 , T_6 , C_3 , C_7):
Not found by port-based classes (Exc.: with SYN-flag rule).
- Conclusion: clusters are better representative of hosts than “port-based” classes

Cross-validation of the classification with BLINC

- Percentage of hosts in (some) clusters falling into the different classes of the BLINC classifier (on transport-level layer)

Id	WEB	UNKN	P2P	MAIL	DNS	FTP	SCAN	CHAT	#Hosts
T_1	60.88	22.04	15.03	0.36	0.86	0.72	0.02	0.08	11637
T_2	7.40	89.95	1.33	0.92	0.27	0.14	0.00	0.00	6344
T_3	8.29	62.10	27.27	0.60	0.67	1.00	0.00	0.07	1626
C_1	90.82	1.18	3.05	2.74	2.14	0.07	0.00	0.00	7875
C_2	90.81	4.10	3.15	0.63	0.75	0.47	0.00	0.04	2765
C_3	15.59	50.61	5.67	2.83	25.10	0.20	0.00	0.00	524
C_4	89.75	5.40	3.37	1.02	0.23	0.23	0.00	0.00	2389
S_1	89.86	3.51	3.30	2.12	0.75	0.16	0.00	0.29	5383
S_6	95.02	4.03	0.58	0.20	0.13	0.05	0.00	0.00	4225
S_7	78.50	18.72	1.80	0.55	0.39	0.04	0.00	0.00	4056
P_1	51.47	1.11	25.40	13.71	6.76	1.45	0.00	0.10	4461

- UNKN is for Unknown.

Comments: Cross-validation with BLINC

- Many UNKN as outputs of BLINC
 - ok in T_1 : web+ P2P receiver; not ok for T_2 : idem for senders
 - problem for BLINC: lack of bidirectionality and of payload in the traces
- MST-based method: separates servers from clients → good for functional-level view of the hosts.
- Cluster C_3 : many anomalous traffic ((SYN and Ping flooding, and on DNS), grouped together by MST-based → an advantage of unsupervised classifier
- Conclusion: satisfactory cross-validations with some improvements

Summary & Perspectives

- Unsupervised classifier for backbone traffic: 9D-feature space + MST-based clustering
- Computation load: takes less than real-time
 - extraction of the clusters requires to process a large enough set of data
 - classification of new traffic = computing the 9 features for each host: very low cost, obtained at the end of the the analyzing window.
 - remember: tested on 15min traces
- Future developments: automation of parameter tuning
- Integration with port-based classifier + anomaly detection + BLINC for automation of cluster labelling

–Further information–

`perso.ens-lyon.fr/pierre.borgnat`