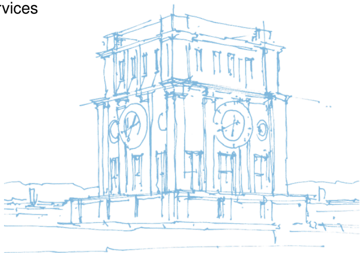
Large-Scale Classification of IPv6-IPv4 Siblings with Variable Clock Skew

Quirin Scheitle, Oliver Gasser, Minoo Rouhi and Georg Carle

TMA'17, Dublin
June 21, 2017

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich



Introduction & Motivation

Motivation

- Increasing IPv6 deployment [1, 2]
- Service-level insights
  $\rightarrow$ Performance comparison, correlated failures and security loopholes [3]
- Research based on initial results of Beverly and Berger [1]
- **Sibling**: IPv6 and IPv4 address pair assigned to the same physical machine [1]

Research question

- Can TCP Timestamp fingerprinting be used to identify siblings?
- Can we identify other metrics significant in decision making?
- How to optimize the decision algorithm (e.g. over-fitting problem)?

Methodology – Overview

- Collect a diverse and large ground truth data set exceeding prior work
- Active measurements against the ground truth
  $\rightarrow$ Parallel full TCP connections to siblings over 10h
- Extract multitude of features from the measurements to discern siblings
- Develop sibling decision algorithms (manual and machine learning) based on the obtained features
- Train and evaluate the algorithms based on train/test split and machine learning

Methodology – Data Sets

- Software, hardware and administrative diversity
- Geographic dispersity
- Various clock characteristics

Data Set	Hosts	#AS	#CC	Skew	Div.
2016-03 (*“03”*)	458	373	40	variable	sw+hw
2016-12 (*“12”*)	682	536	80	variable	sw+hw
servers	31	9	5	variable	sw+hw
ring	430	383	56	variable	hw
RAv1	12	12	11	variable	-
RAv2	209	192	64	constant	-
Beverly [1]	61	34	19	constant	unkn.

- Non-siblings created by mixing addresses from different servers

Features

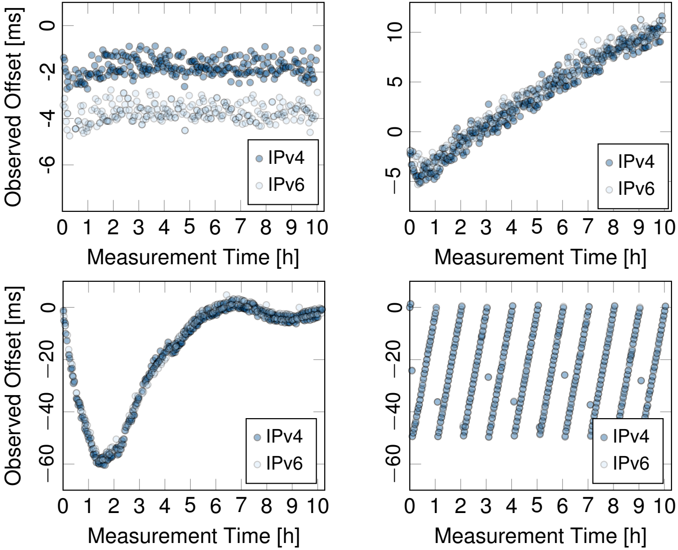
Terminologies

- Falsifying or verifying
- Clock offset: The time difference between the target and local clock
- Clock skew: The frequency difference between the target and the local clock
  $\rightarrow$ First derivative of the offset

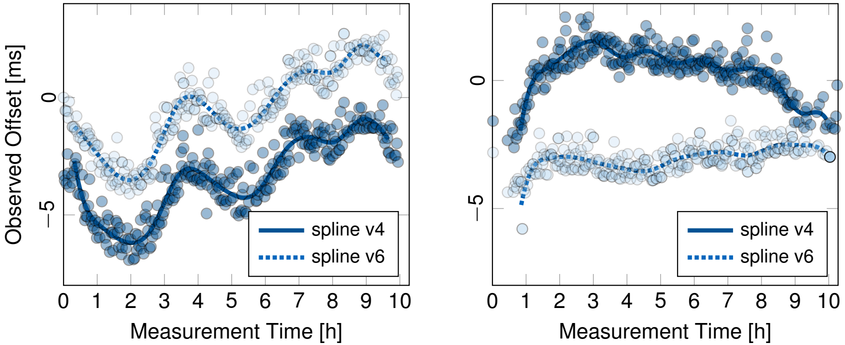
Prominent features

- TCP options fingerprint
  - Presence and order of options + value of Window Scale option
  - More hosts than operating systems $\rightarrow$ only verifying
- Raw TCP timestamps
  - Delta of two TCP timestamps of pair (2^{32} entropy)
  - High discriminative power
- Clock offset and skew related metrics

Classes of Skew Observation



A Classification Example Using Polynomial Splines



Fitted Splines for Sibling (left) and Non-Sibling (right).

Classifier Models

- Hand-tuned decision algorithm
- Machine learning approach

Classifier Models – Hand-tuned Decision Algorithm

- Manually tune the algorithm based on extracted features
- Hand-tuned on *2016-03* (≈40%) and tested on newly added hosts of *2016-12* (≈60%)

Steps (Simplified)

- Falsify pairs with different TCP options signature
- Falsify based on raw TCP timestamp evaluation
- Detect the skew class
  - Linear testing
  - Negligible skew → Unknown classification
  - Variable skew → Polynomial splines

Classifier Models – Hand-Tuned Algorithm

- Good performance (discussed later)
- Complex algorithm ($>$20 decision points, $>$10 parameters)
- Significant effort to retrain and adjust to grown ground truth data sets
- Only 1 train/test split (no cross-validation) $\rightarrow$ might not generalize well

Applying Machine Learning

- Start with the most simple classifier—Decision Tree
- Conduct rigorous analysis: 10-fold cross-validation on various train/test splits
- Validate sensitivity to groups of hosts in ground truth, e.g., RIPE Atlas, NLNOG Ring by training/testing on these groups
- Employ Matthews Correlation Coefficient (MCC) as stable metric, superseding easily biased metrics such as Precision or Accuracy

Surprising Results!

Machine-Learned Decision Tree achieves MCC of 1.0 (with still very few errors) based on just 1 threshold on the Δ_{tcpraw} metric $\rightarrow$ significant simplification of algorithm!

Table 1: Hand-Tuned and Machine-Learned Classifiers train and test very well, speaking to good generalization.

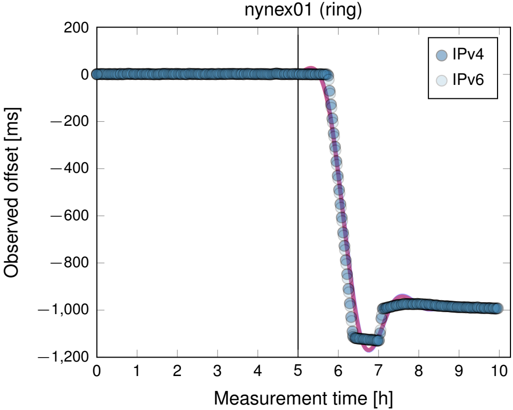
Algo.	Train DS	Test DS	Prec.	MCC	Type
HT1	03	03	100%	.99	Train
HT1	03	$12 \setminus 03$	99.49%	.98	Test
ML1	$03 \cup 12$	$03 \cup 12$	99.36%	1.0	Train
ML1	$03 \cup 12$	$03 \cup 12$	99.88%	1.0	Test

ML1 values are the arithmetic mean of 10-fold cross-validation.

$\rightarrow$ Machine learning did not only help to build a good classifier, but also makes more rigorous analysis, such as cross-validation, easily possible.

Anecdote — 2016 Leap Second

We conducted remote TCP timestamp measurements for 30 hours around the 2016 year end leap second. Remember: The TCP timestamp clock should monotonically tick without interference from the OS clock.



More plots of interesting leap second behavior in paper!

Recent Changes to the Linux Kernel

On May 5, 2017, Linux[1] integrated a patch to randomize TCP timestamps for each source IP address. What does this mean for our method?

- Methods based on raw TCP timestamp value will not work any more
- Methods based on long-term skew will still work
- Wide-scale rollout of Linux 4.10 Kernel will take a while → conduct your studies fast

Reasons for this change in Linux include protection against uptime estimation and NAT device enumeration.

1: `https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=84b114b9`

Future Work

- Drastically reduce 10h measurement time – current work-in-progress algorithm achieves an MCC of $>.95$ with only 1 packet (will be impacted by Linux patch)
- Identify and classify sibling candidates based on passive observations (work-in-progress)
- Investigate impact and possible workarounds of Linux patch more closely
- Integrate more ground-truth hosts – your help needed :)

Key Messages, Data, and Code

- We extend work of Beverly and Berger to support variable clock skew and novel features
- Using machine learning and a significantly larger ground truth, we provide a more simple yet better generalizing model
- We apply our methodology to 8.9M sibling candidates
- We open-source our code, ground truth, data, and detailed results

Data and Code:
`https://github.com/tumi8/siblings`

Quirin Scheitle, Oliver Gasser, Minoo Rouhi, and Georg Carle
Chair of Network Architectures and Services — `https://net.in.tum.de`
Technical University of Munich (TUM)

Bibliography

[1] R. Beverly and A. Berger.
Server Siblings: Identifying Shared IPv4/IPv6 Infrastructure via Active Fingerprinting.
In *International Conference on Passive and Active Network Measurement*, pages 149–161. Springer, 2015.

[2] R. Beverly, L. Campbell, A. Berger, and N. Weaver.
Inferring Internet Server IPv4 and IPv6 Address Relationships.
Technical report, Monterey, California: Naval Postgraduate School, 2013.

[3] J. Czyz, M. Luckie, M. Allman, and M. Bailey.
Don't Forget to Lock the Back Door! A Characterization of IPv6 Network Security Policy.
In *Network and Distributed Systems Security (NDSS)*, Feb 2016.

[4] V. Paxson.
End-to-end Internet packet dynamics.
In *ACM SIGCOMM Computer Communication Review*, volume 27, pages 139–152. ACM, 1997.

[5] Rouhi Vejdani, Minoo.
Path Tracing and Validation of IPv4 and IPv6 Siblings.
Master's thesis, Technische Universität München, 2016.

[6] Q. Scheitle, O. Gasser, M. Rouhi, and G. Carle.
Large-Scale Classification of IPv6-IPv4 Siblings with Variable Clock Skew.
In *Network Traffic Measurement and Analysis Conference (TMA)*, Dublin, Ireland, June 2017.