UNIVERSITY OF CAMBRIDGE

Yet Another Heavy Hitter Detection Problem [on going work]

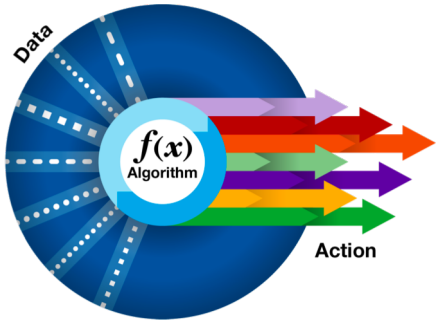
Salvator Galea, Gianni Antichi, Andrew W. Moore

Department of Computer Science and Technology

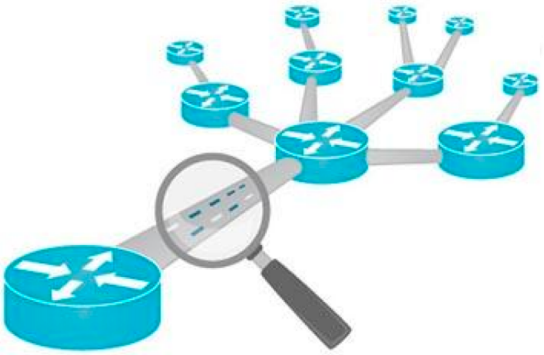
NEC \Orchestrating a brighter world

Network Management

Monitors Network Traffic



Applies Management Techniques



Measures Network Traffic

Measurements as support for Management

- Form the basis of all traffic management functions
- Complex set of data traffic need to be analysed.

- The analysis of the results will trigger the network tasks that need to be applied.
  - Better traffic engineering techniques
  - Better Quality of Service
  - Better security

How do we analyse this complex set o data?

Do we need to classify the traffic?

Flows are important

Providing an aggregate view of such data is important for summarization, visualization, and analysis.

Flows represent a number of packets or frames passing a network point during a certain time interval. The packets, which belong to a flow, have a set of common properties.

- Traffic classification types (bursty, latency-sensitive, traffic-pattern change)
- Apply management tasks (QoS, capacity planning, efficient traffic engineering)

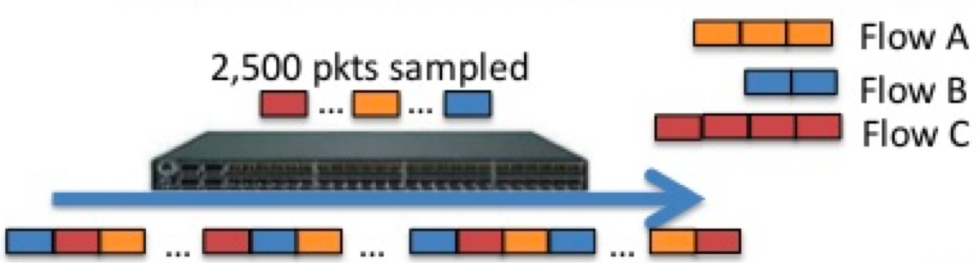
Ex. Flows with high volume of traffic (a.k.a **Heavy Flows**) are interesting!
Management task : traffic engineering, accounting

Ideal Measuring Tool vs Reality

Unlimited resources (to store all the counters) + Fast traffic statistics processing

BUT

Resources in the observation point usually have a fixed size that is either constrained by hardware.



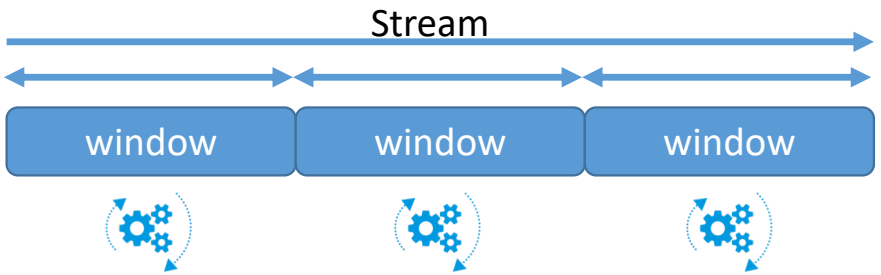
Solutions with acceptable(?) less accuracy:

- packet sampling
- streaming algorithms

"If we're keeping per-flow state, we have a scaling problem, and we'll be tracking millions of ants to track a few elephants." — Van Jacobson, End-to-end Research meeting, June 2000.

Streaming Algorithms

Sliding window model



Streaming traffic divided into multiple fixed time windows.

Why in windows?

- Easy to implement. Feasible observation and collection of statistics.
- Prevent counters overflow by flushing

So far so good ehm, so what's the pitfall?

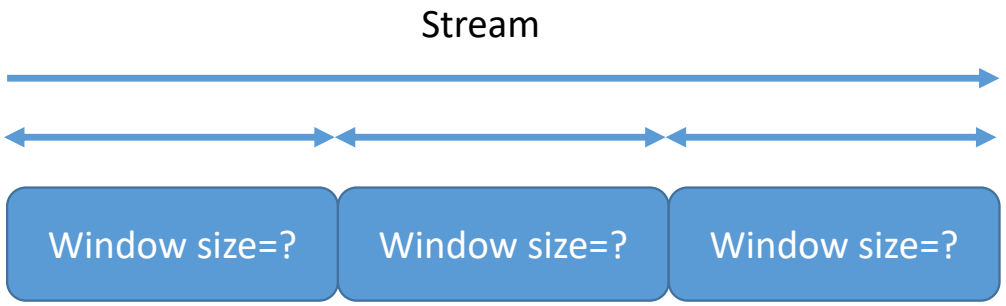
- At the end of each time-interval, collect flow statistics and flush the counters
- This create a coupling between detection and the window size itself

Window Size

Is the window size a problem?

What's the "right" window size?

- Datacenters
- ISP Backbones



Generic question: Do small variations of the window size affects the traffic statistics?

Window size = 10sec
10±1sec
10±1msec
10 ±1usec

Same results?

Experimental setup

Offline Analysis Tool*

Prefixes	: Source IP
Baseline	: 10sec window
Threshold	: 5% of the total traffic in the window
Comparison metric	: Jaccard similarity coefficient
	(used for comparing the similarity and diversity of sample sets.)
Traces	: CAIDA2016 DirA(~40M Packets and ~180K Flows for 20min traces)
Detection	: Heavy Hitters, Hierarchical Heavy Hitter, Leaf Heavy Hitters, Top-k flows

Lets experiment and see…

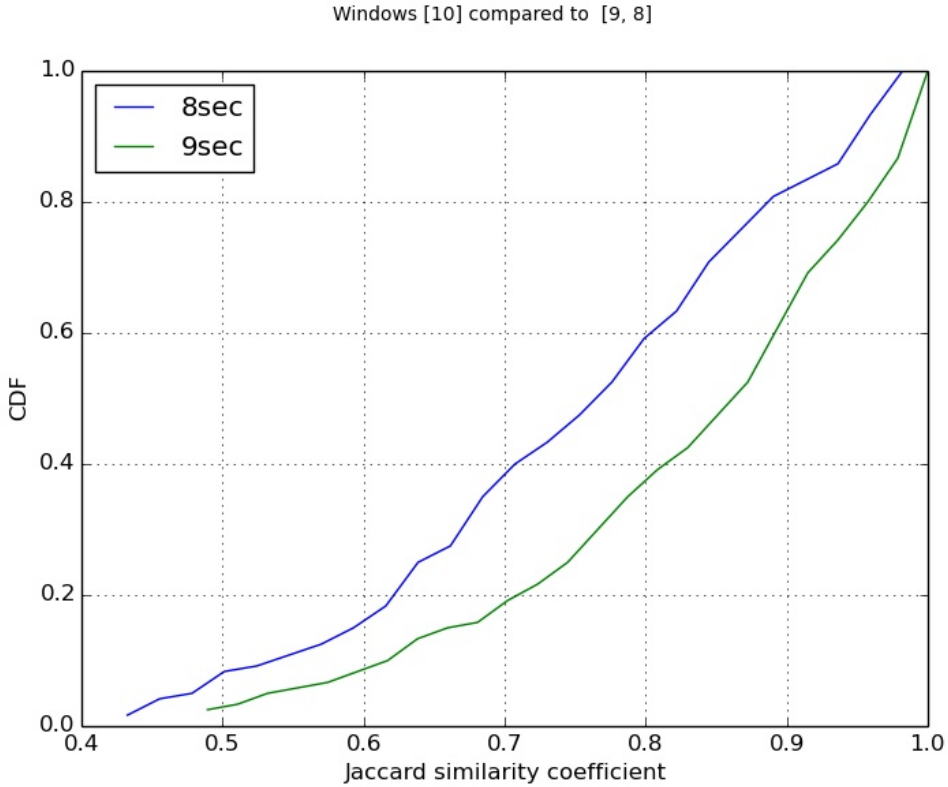
*Acknowledgment : Jan Kucera (analyzer tool)

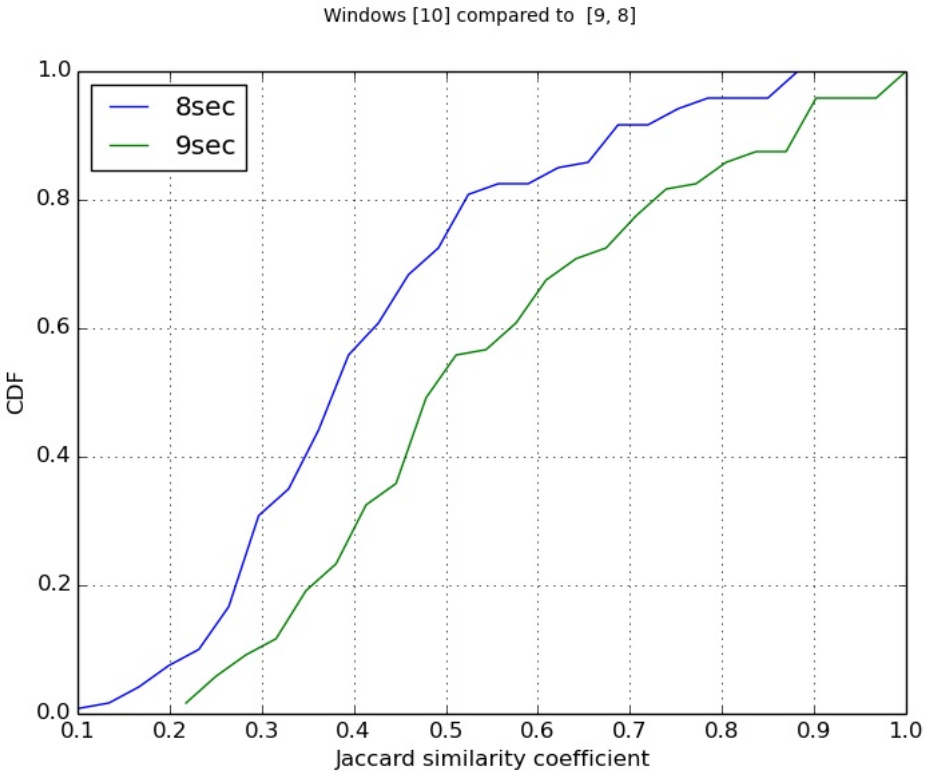
Offline experiments

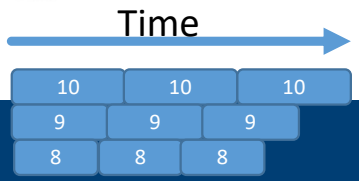
Details

	Detection	Trace	Windows Comparison (sec)
Test 1	HH,HHH	20 minutes	[10] to [9,7]
Test 2	HH,HHH	10 minutes	[10] to [9.9, 9.8, 9.7]
Test 3	HH,HHH	20 minutes	[10] to [9.99, 9.96, 9.93, 9.90]
Test 4	HH, HHH	20 minutes	[10] to [10]+offset[1, 2, 3]
Test 5	LeafHH, Top-k Flows	60 minutes	[10] to [10]+skip_start[1, 2, 3, 4]

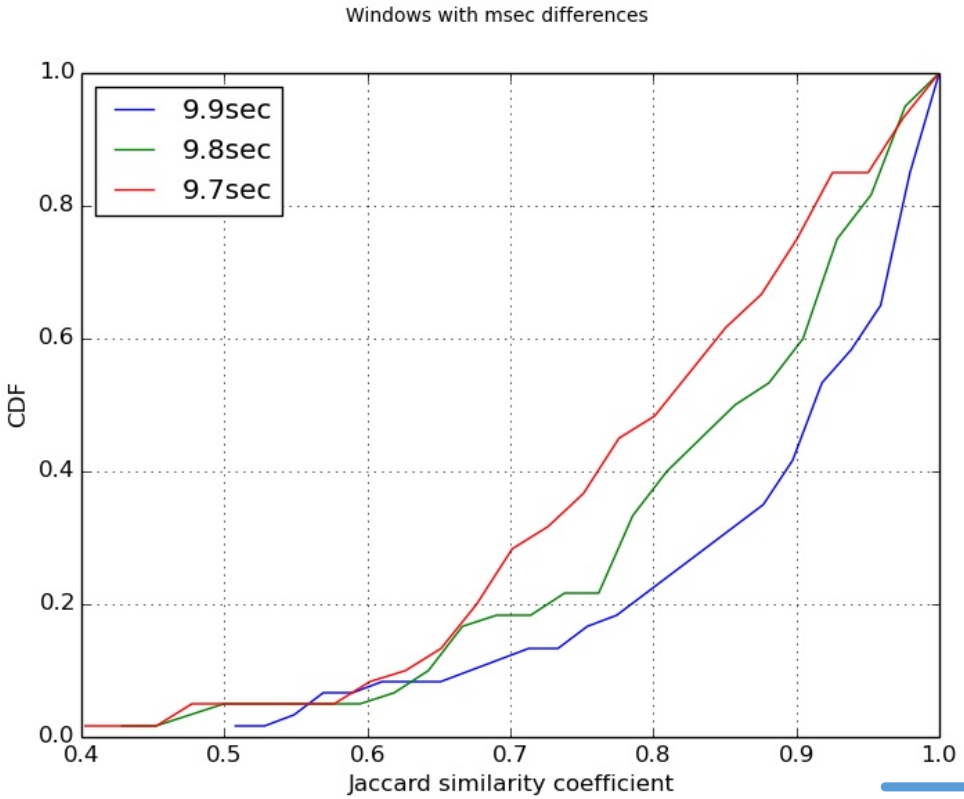
Experiment 1 (HH + HHH), sec

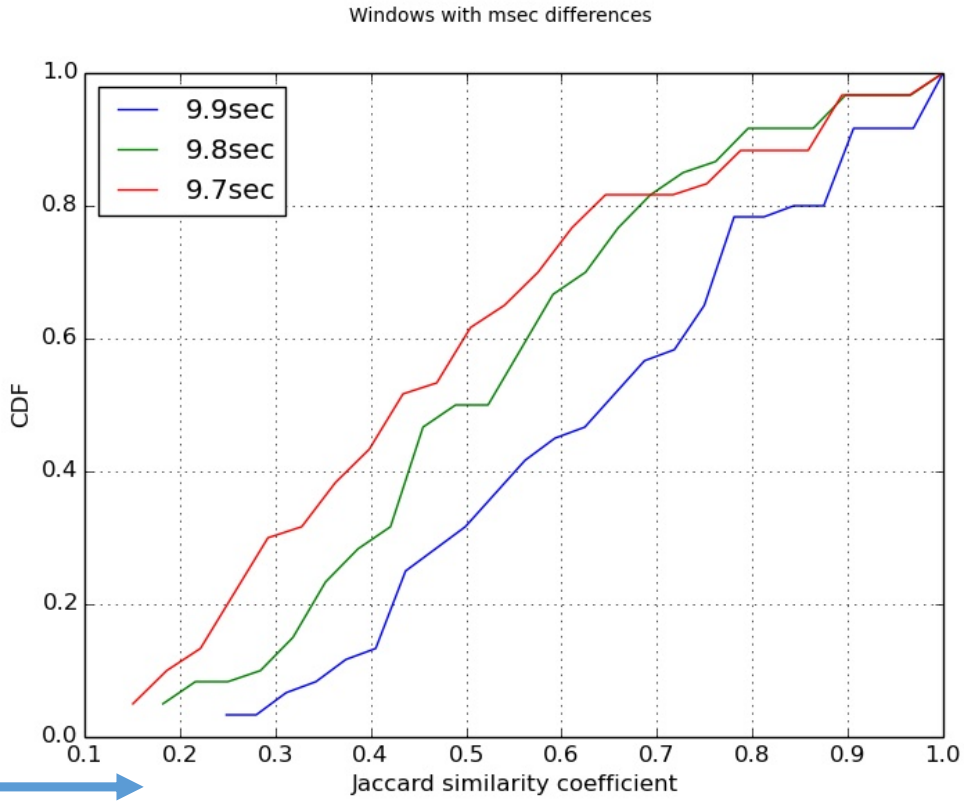


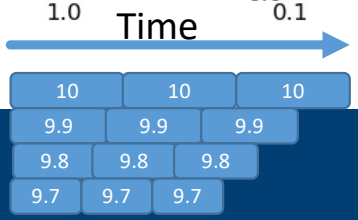




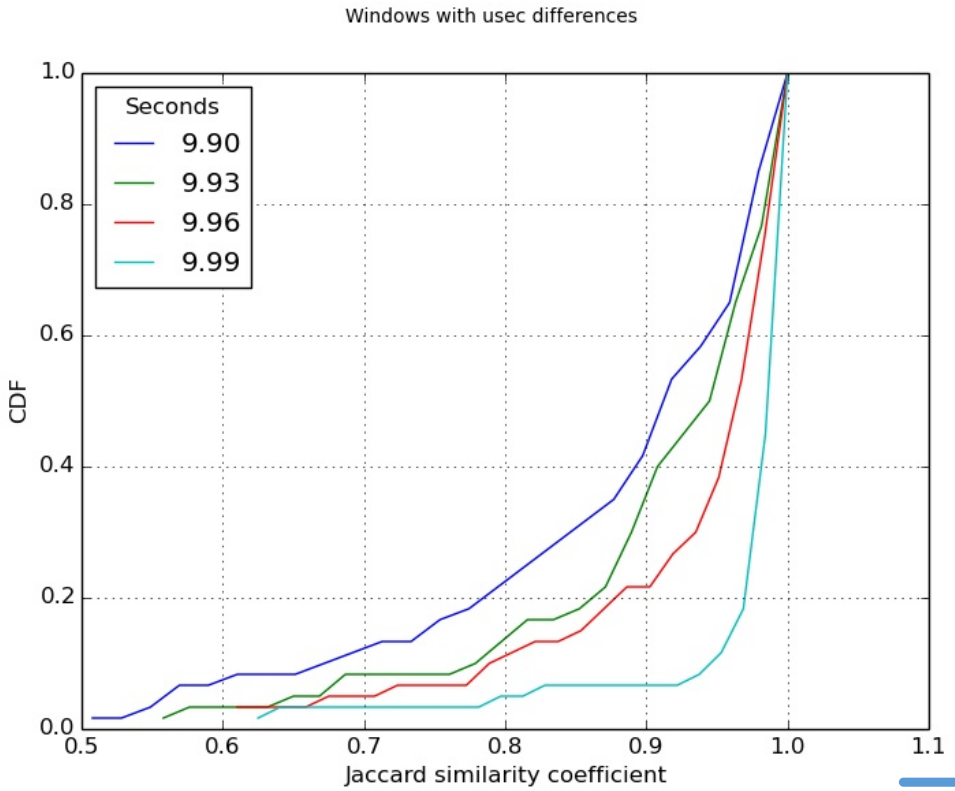
Experiment 2 (HH + HHH), msec

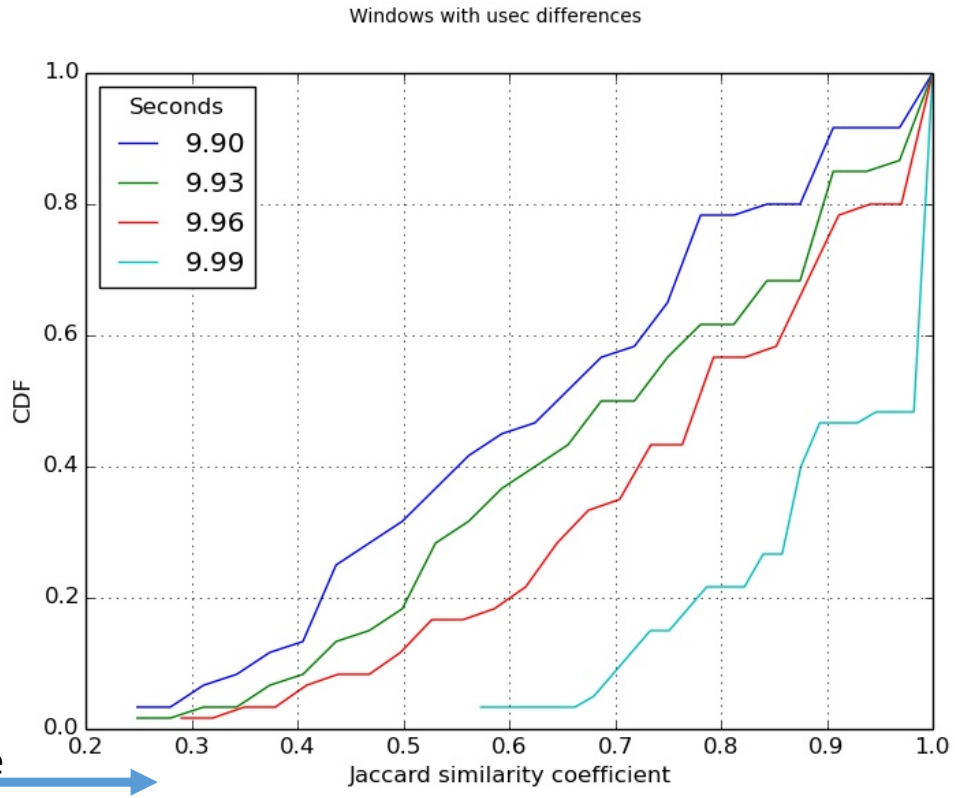






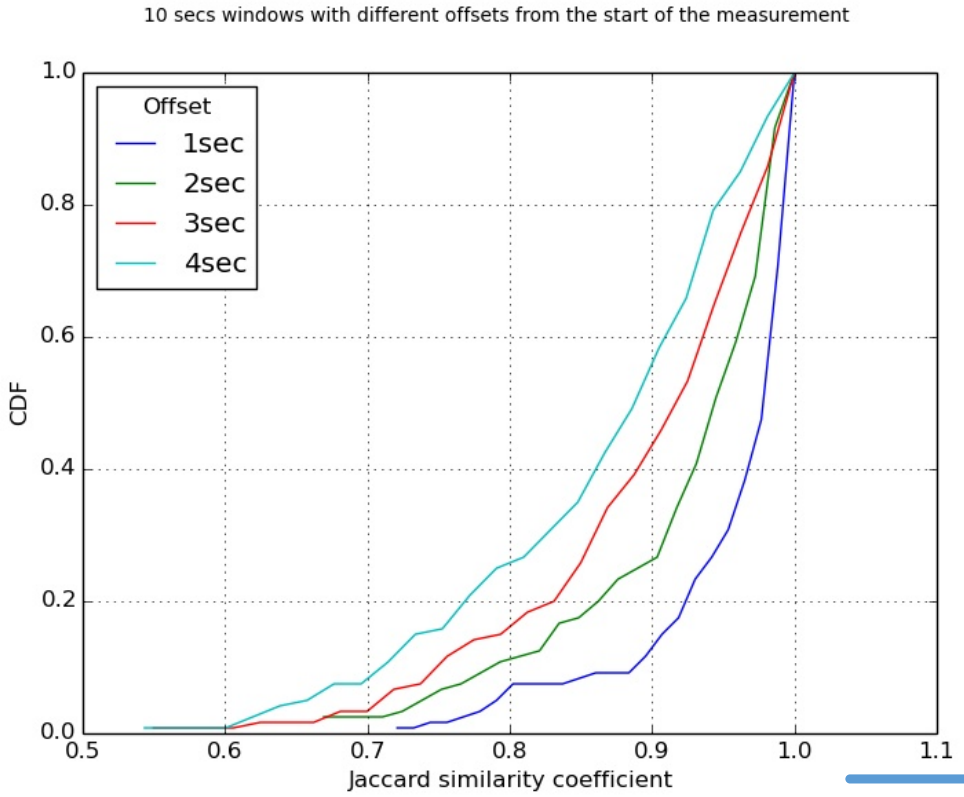
Experiment 3 (HH + HHH), usec

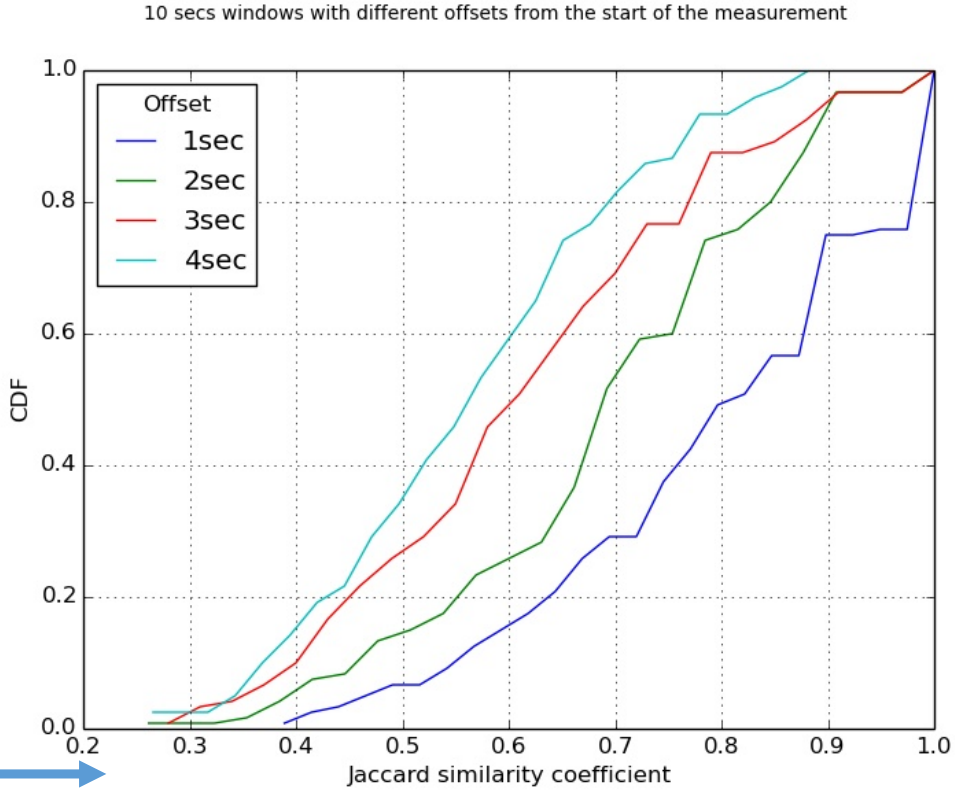


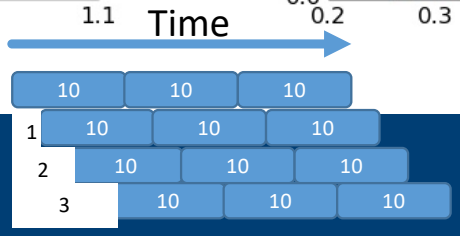


UNIVERSITY OF CAMBRIDGE

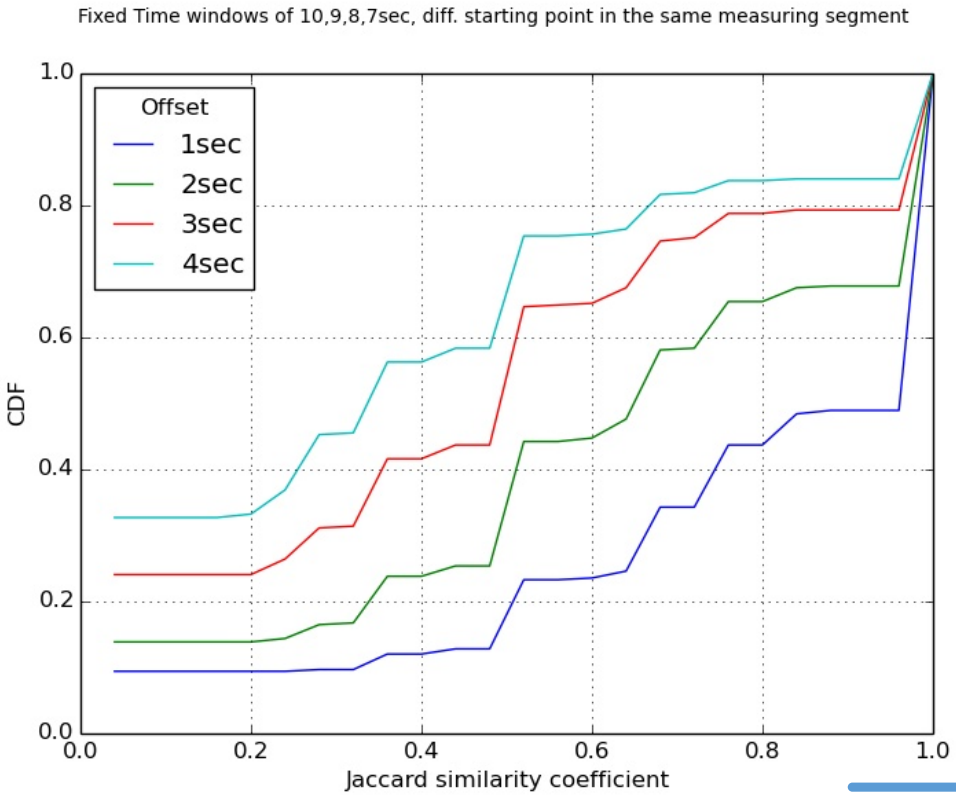
Experiment 4 (HH + HHH), +offset

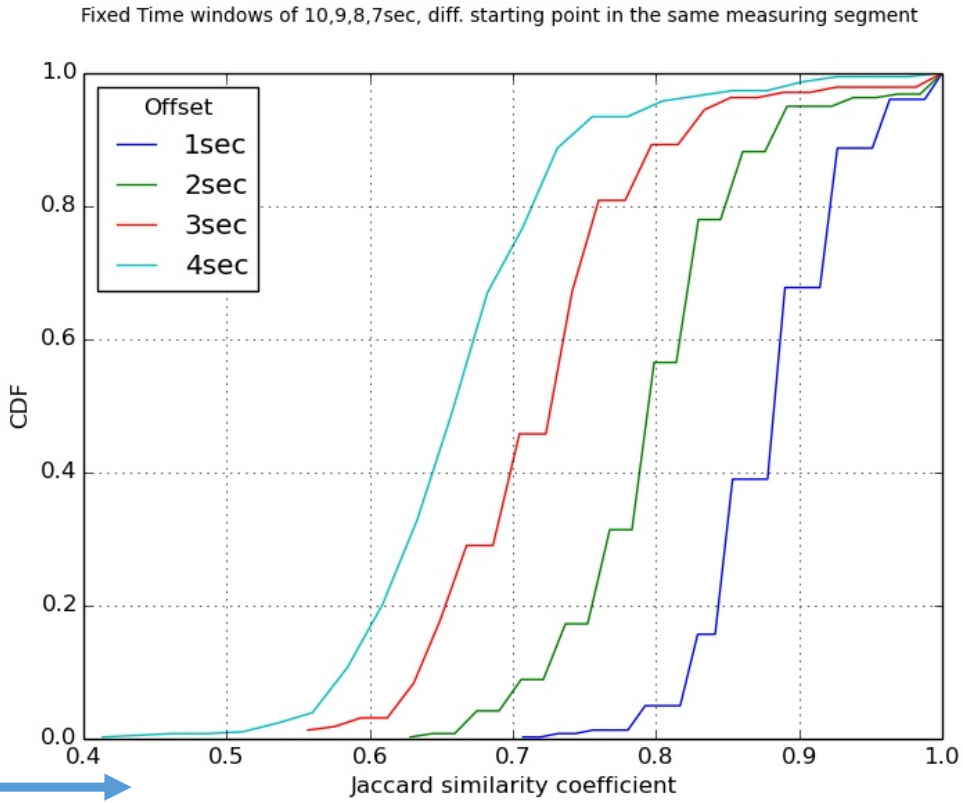


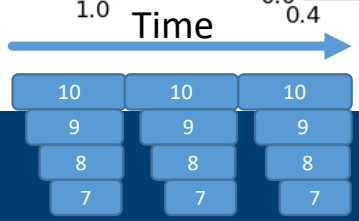




Experiment 5 (LeafHH + Top50 Flows)







So what? Food for thought

Why is this happening?

What's the impact that this can have?

Even the small differences in the window size give different perception of Heavy Flows

Different time windows or different starting points of the same time window can produce different statistics.

Bloom Filters + Future Work

Counting Bloom Filter

Probabilistic data structure which maintains the frequency count for each item in a data stream

Window-based with Time-Decaying Counters

The value of each counter decays with time, by applying exponential time-decaying function

The significance of data items decreases over time

Continuous-Time Decaying Counters

On-demand Time-decaying Bloom filter, which relies on a continuous-time operation to overcome the accuracy/performance limitations of the original window-based approach

Any suggestions?

Any question?

Reference Kai Cheng, Limin Xiang and M. Iwaihara, "**Time-decaying Bloom Filters for data streams with skewed distributions**," *15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications (RIDE-SDMA'05)*, 2005
Reference: Giuseppe Bianchi, Nico d'Heureuse, and Saverio Niccolini. 2011. **On-demand time-decaying bloom filters for telemarketer detection**. SIGCOMM Comput. Commun. Rev. 41, 5 (October 2011)