Estimating the number of active flows in a data stream over a sliding window

Éric Fusy and Frédéric Giroire

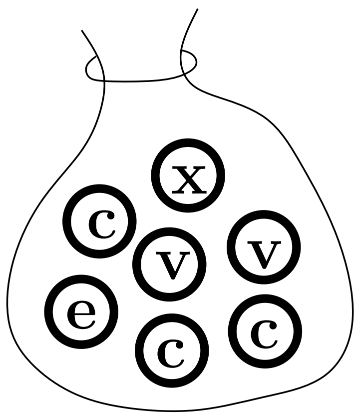
Algorithms Project, INRIA Rocquencourt

Overview

Estimation of large cardinalities

Cardinality of a multiset

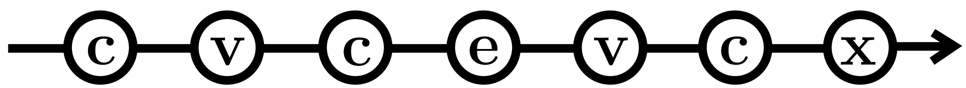
- Let $\mathcal{M}$ be a multiset,
  - N is the number of elements called the *size*
  - n the number of distinct elements called *cardinality*.

$\mathcal{M} =$ 

size $N = 7$

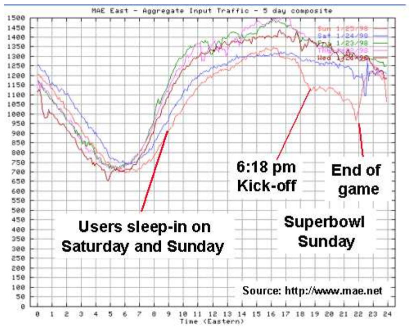
cardinality $n = 4$

elt.	e	v	c	x
mult.	**1**	**2**	**3**	**1**



- **Problem:** compute the cardinality n in one pass and with small auxiliary memory.

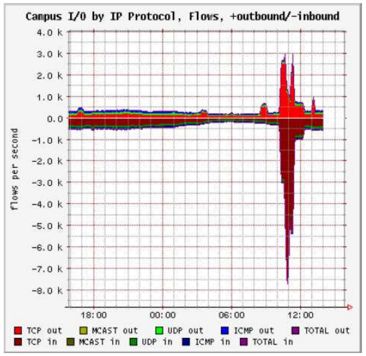
Surprisingly long list of applications



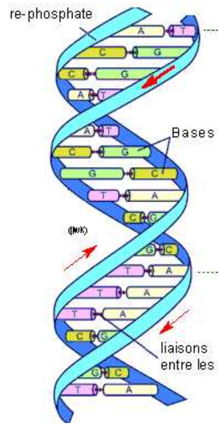
Traffic analysis

Linguistic

Very large multisets!



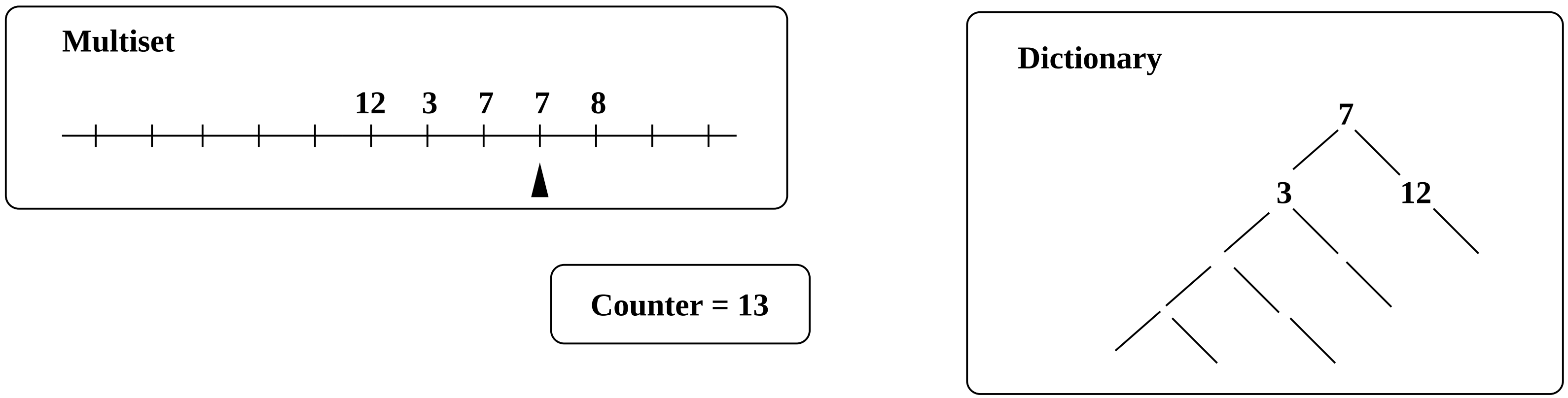
Detection of attacks



Analysis of the genome

Exact solution

- Maintain distincts elements already seen.



- One pass, but auxiliary memory of order n.
- **Information theory:** memory $\Omega(n)$ necessary

Probabilistic solution

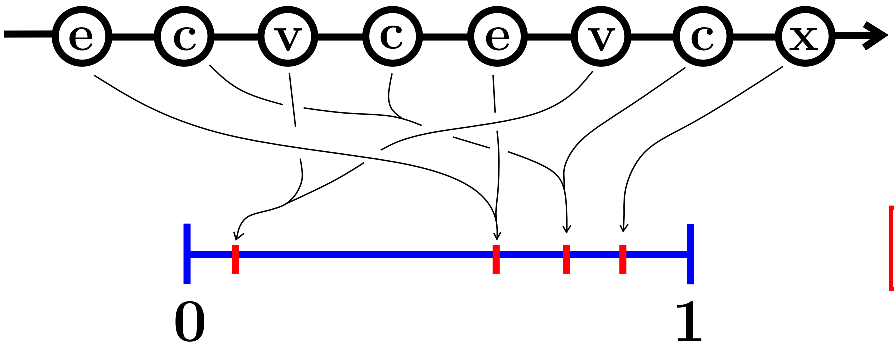
Crucial idea: relax the constraint of exact value of the cardinality. An estimate with good precision is sufficient for the applications.

Several algorithms have been proposed

- *Probabilistic Counting*, Flajolet and Martin 1983.
  LogLog Counting, Durand and Flajolet 2003.
- *Linear Counting*, Whang, Zanden and Taylor 1990.
- *Counting distinct elements in a data stream*, Bar-Yossef et al. 2002.

Probabilistic solution

- Elements of $\mathcal{M}$ are hached to $[0,1]$.



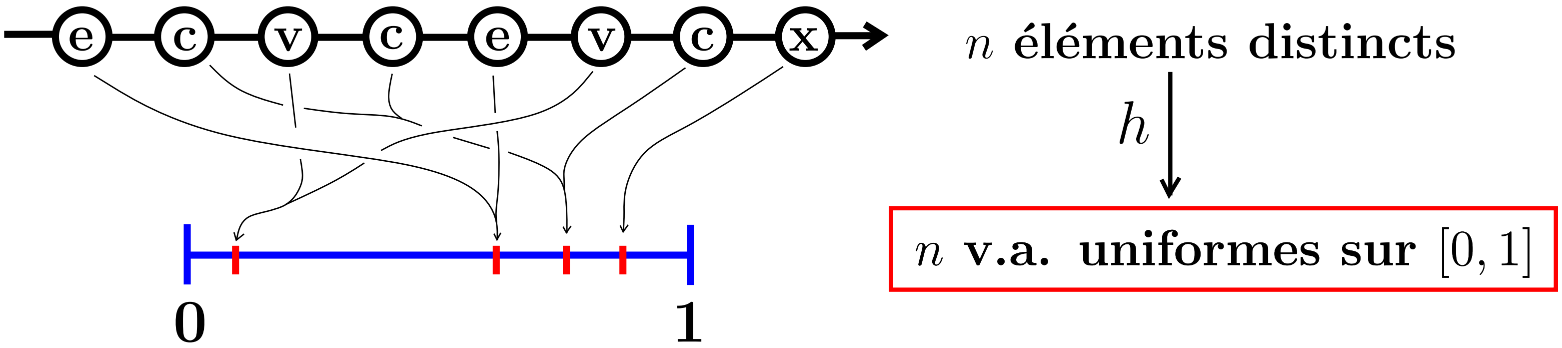
n **éléments distincts**

$$\Big\downarrow h$$

$$\boxed{n \textbf{ v.a. uniformes sur } [0,1]}$$

Probabilistic solution

- Elements of $\mathcal{M}$ are hached to $[0,1]$.

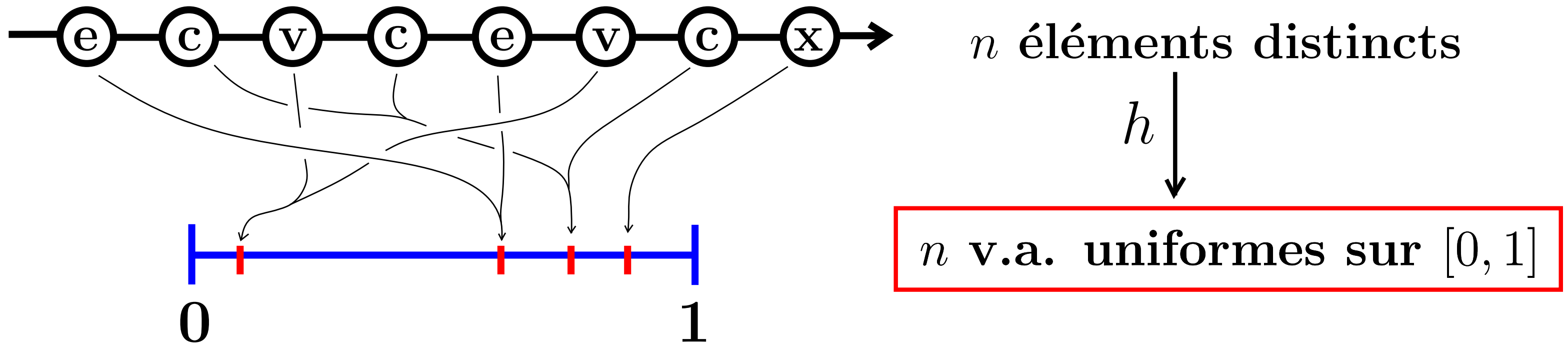


$$\mathbb{E}(\mathrm{Min}) = \frac{1}{n+1}$$

$\Rightarrow$ Idea: use the minimum to estimate the cardinality

Probabilistic solution

- Elements of $\mathcal{M}$ are hached to $[0, 1]$.



$$\mathbb{E}(\mathrm{Min}) = \frac{1}{n+1}$$

$\Rightarrow$ Idea: use the minimum to estimate the cardinality

- The minimum is computed in one pass with constant memory

The algorithm MinCount

- Simulate m hashing functions
  $\Rightarrow m$ minima $M^{(1)}, \ldots, M^{(m)}$
- Estimate $= \alpha_m \times$ geometric mean of the $1/M^{(i)}$
- Relative error $\approx 1/\sqrt{m}$ for a memory of m words
  Accuracy of 4% with only 1kB of memory!
- If some buckets are empty (no minimum) use the number of empty buckets to estimate the cardinality

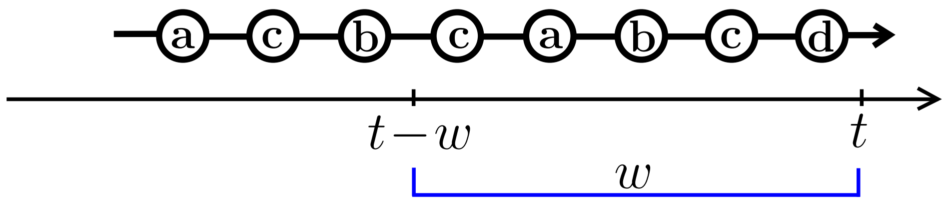
Counting over a sliding window

New context

- Telecom context: stream of IP packets passing by a router
- Each packet belongs to a flow (connection), identified by <source IP, destination IP>
- Elements of the multiset = packets
  Distinct elements of the multiset = flows
- Typical request: "What is the number of active flows over the last hour"

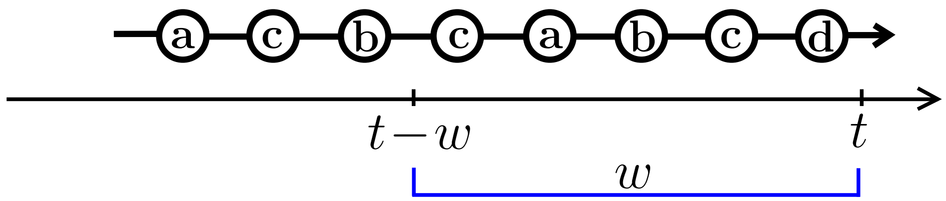
Sliding window

- Model studied by [Datar, Gionis, Indyk, Motwani]: "Maintaining Stream Statistics over a Sliding Window"
- Problem: At each time t, we want to estimate the number of flows over $[t - w, t]$.



Sliding window

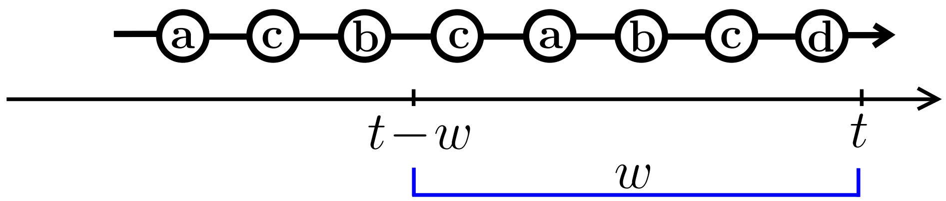
- Model studied by [Datar, Gionis, Indyk, Motwani]: "Maintaining Stream Statistics over a Sliding Window"
- Problem: At each time t, we want to estimate the number of flows over $[t - w, t]$.



- Our contribution (ANALCO'07):
  - New algorithm Sliding MinCount extends the (static) MinCount algorithm to the sliding window model
  - Complete analysis of the auxiliary memory
  - Validation on real traffic.

Sliding window

- Model studied by [Datar, Gionis, Indyk, Motwani]: "Maintaining Stream Statistics over a Sliding Window"
- Problem: At each time t, we want to estimate the number of flows over $[t-w, t]$.



- Our contribution (ANALCO'07):
  - New algorithm Sliding MinCount extends the (static) MinCount algorithm to the sliding window model
  - Complete analysis of the auxiliary memory
  - Validation on real traffic.

The approach

- MinCount uses an estimate based on the minimum
- We have to maintain the minima of hashed values over a sliding window
- **Difficulty**: over a sliding window, outdated elements are discarded

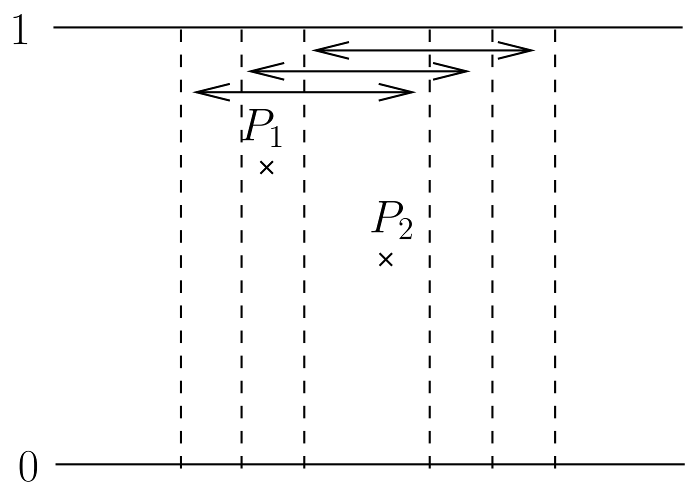
"How to remember if the discarded element has realized the minimum or not?!"

Maintain the minimum

- Solution: keep in memory the packets that may become a minimum in the future
- Crucial remark If $P_1 = (h_1, t_1)$ and $P_2 = (h_2, t_2)$ are two packets such that

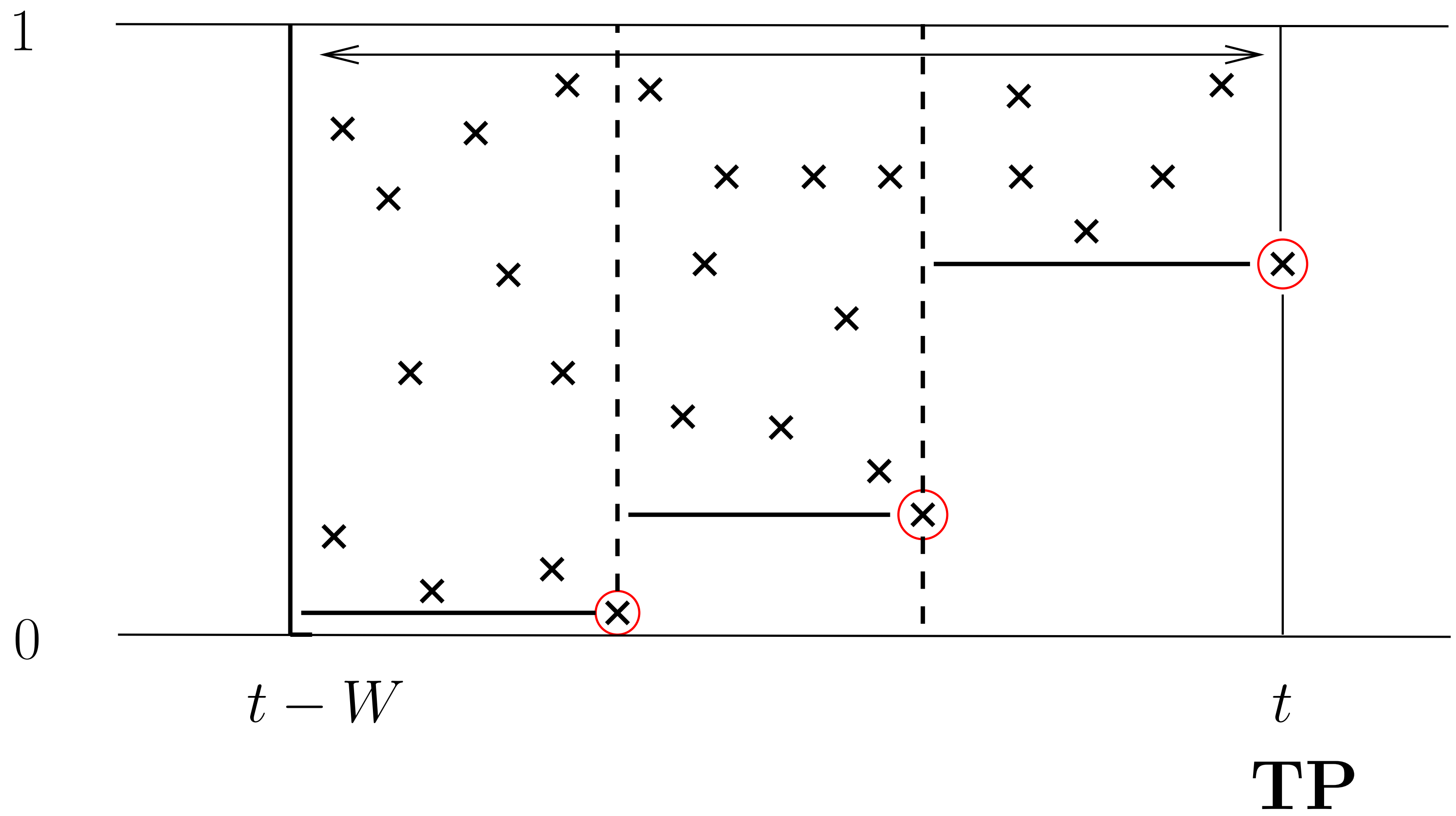
$$t_1 < t_2 \quad \text{and} \quad p_1 \geq p_2,$$

then P_1 can not become a minimum in the future.



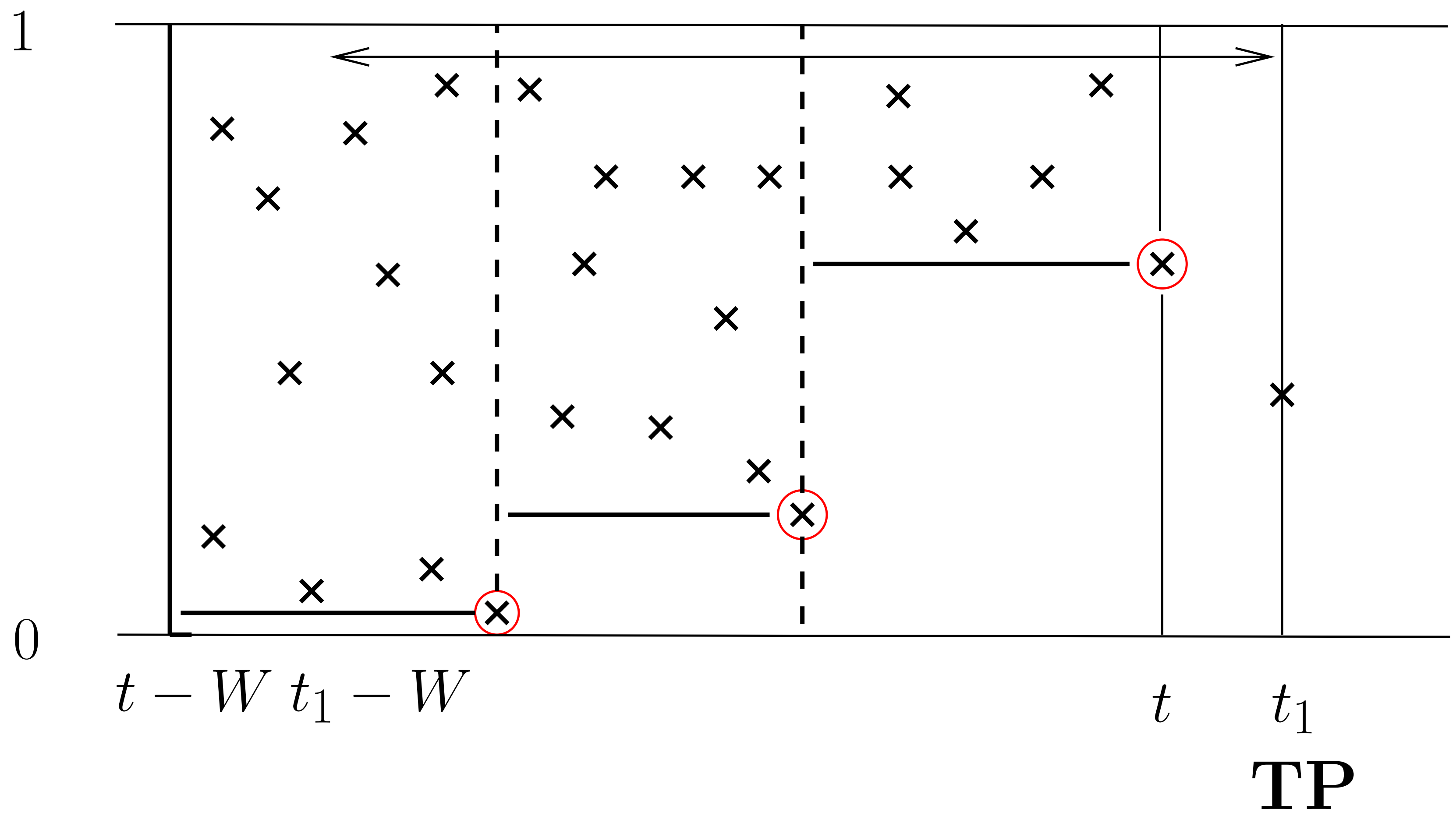
The list of future possible minima

The future possible minima are the minimal records of hashed values taken in reverse chronological order.



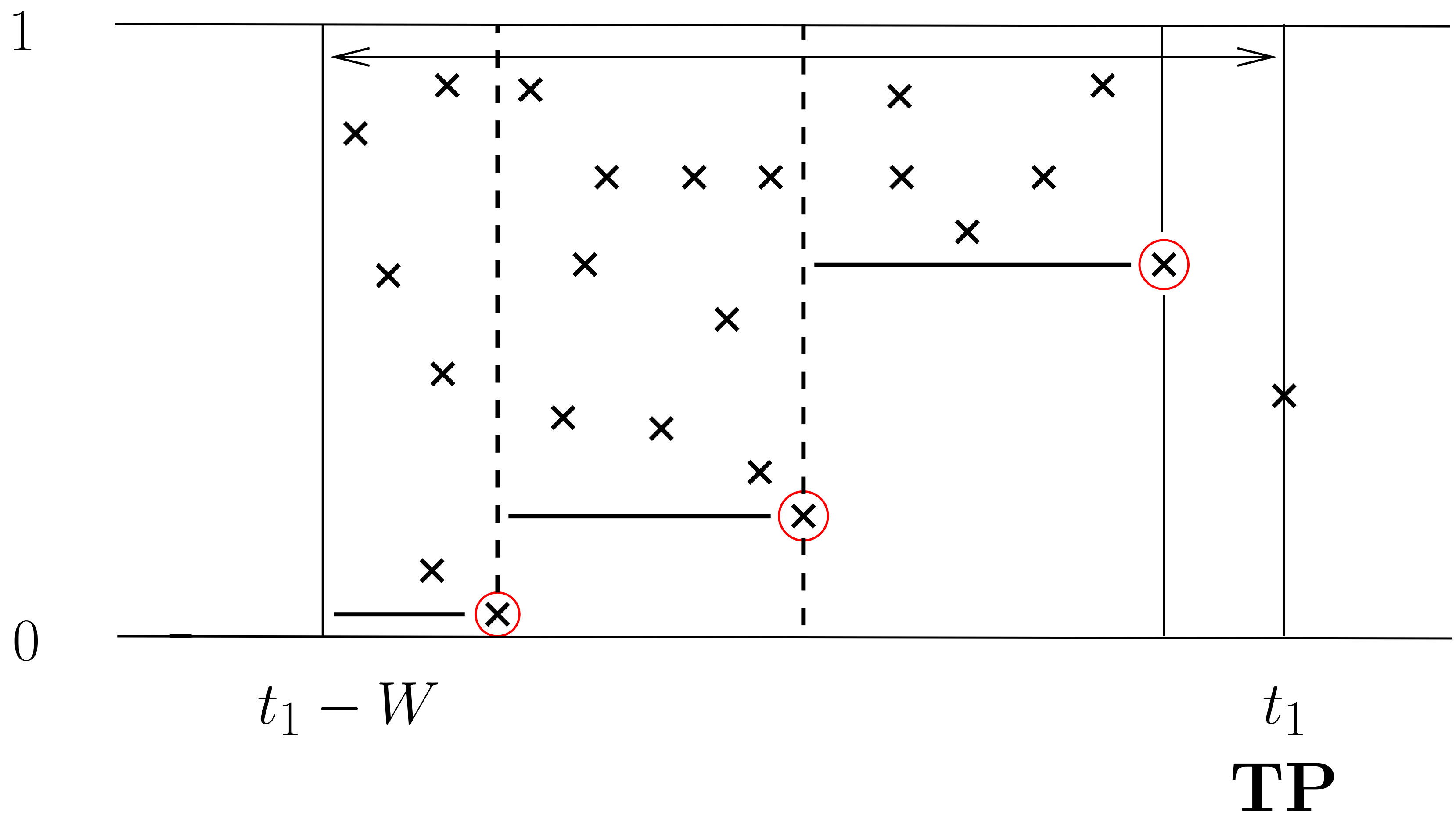
The list of future possible minima

The future possible minima are the minimal records of hashed values taken in reverse chronological order.



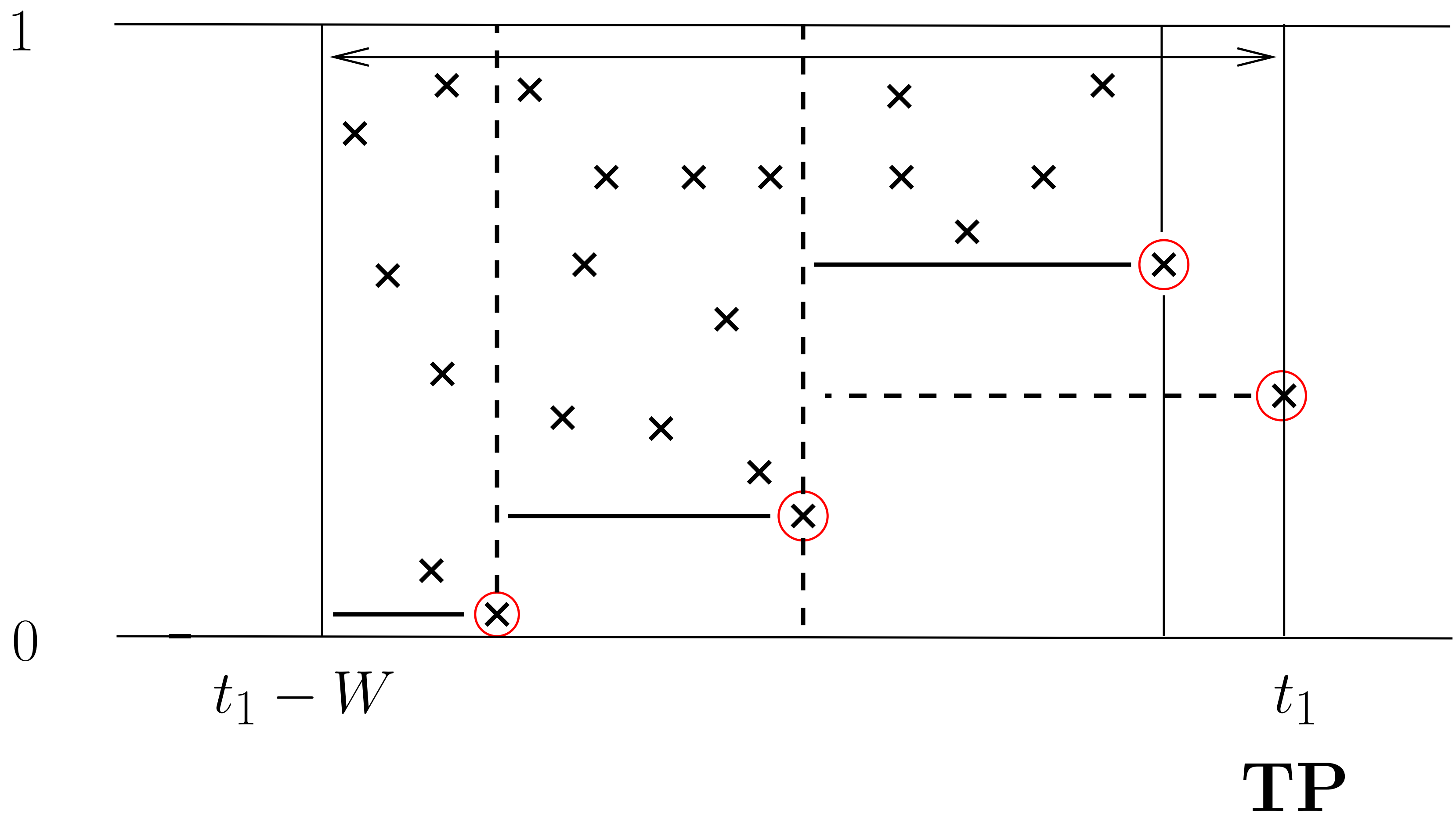
The list of future possible minima

The future possible minima are the minimal records of hashed values taken in reverse chronological order.



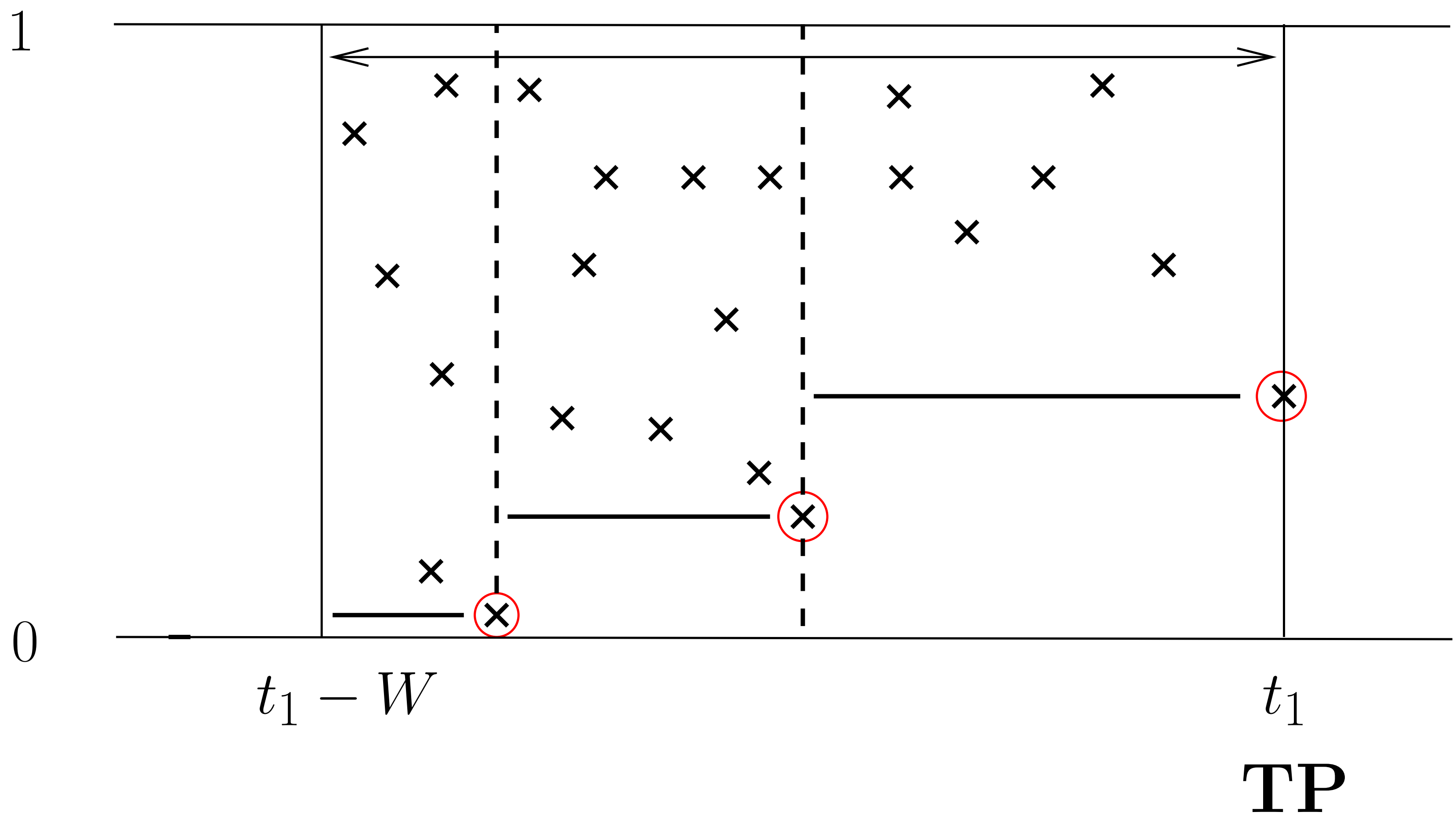
The list of future possible minima

The future possible minima are the minimal records of hashed values taken in reverse chronological order.



The list of future possible minima

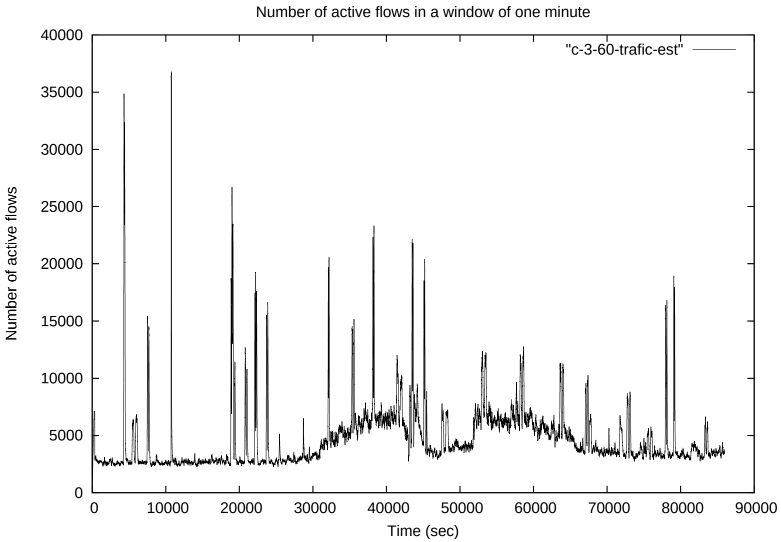
The future possible minima are the minimal records of hashed values taken in reverse chronological order.

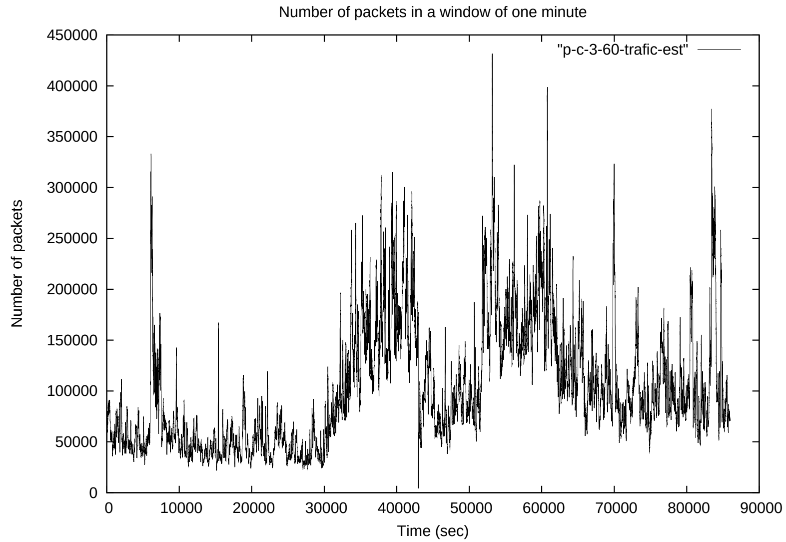


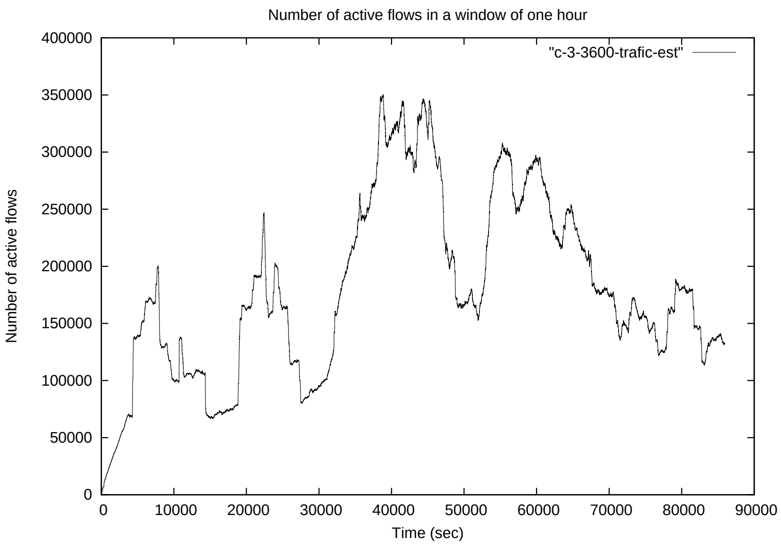
Results

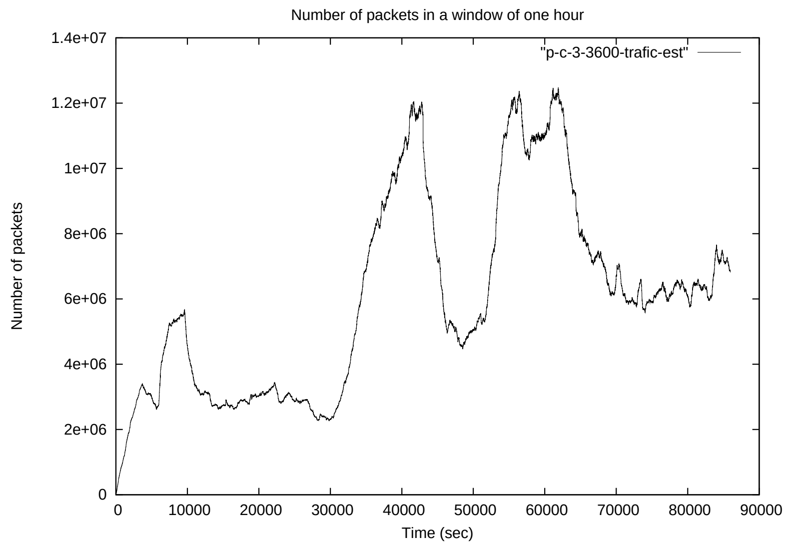
- Same accuracy as MinCount

Application to traffic monitoring

- 







Aggregated traffic of 400 machines at INRIA during one day. Comparison number of flows (Left) / number of packets (Right), for window of 1 hour (Top) / 1 minute (Bottom).