Clock Phase Change compensation using Graham Scan

Augusto Ciuffoletti
Università di Pisa

Galina Antonova
GE Consumer & Industrial, Multilin

Problem statement

- Ensuring high timing accuracy for Slave clocks built with cheap oscillators requires frequent updates from the Master clock.
- Clock Phase Change linear compensation may significantly reduce updates’ frequency, while maintaining required timing accuracy.
- Temperature and aging (non linear) components of the Clock Phase Change remain to be compensated.

Graham Scan requirements

- The algorithm requires that a Slave clock receives series of timestamped messages from a Master.
- Although the message flow should be regular, no strict timeliness is required.
- Such messages do not need special privileges, but regularity in the delivery helps.

Probabilistic issues

- The algorithm makes (weak) hypotheses on the distribution of message latency.
- The algorithm returns an <u>estimate</u> of the Clock Phase Change, not the “real” value.
- Accuracy of such estimate improves for longer series of messages.
- The algorithm improves performance of a clock synchronization algorithm, but does not replace it.

Advantages

- Low cost/impact algorithm.
- Adequate to a wireless environment: Slave does not transmit thus saves power.
- Adequate to a broadcast/multicast environment: one message serves multiple Slaves.
- May be sufficient (no additional clock synchronization needed) for applications that only need to measure time intervals (e.g. monitoring, accounting, debugging).

Basic notation

$$ts(rcv_i) - ts(snd_i) = \Delta_i + a * ts(snd_i) + b$$

where

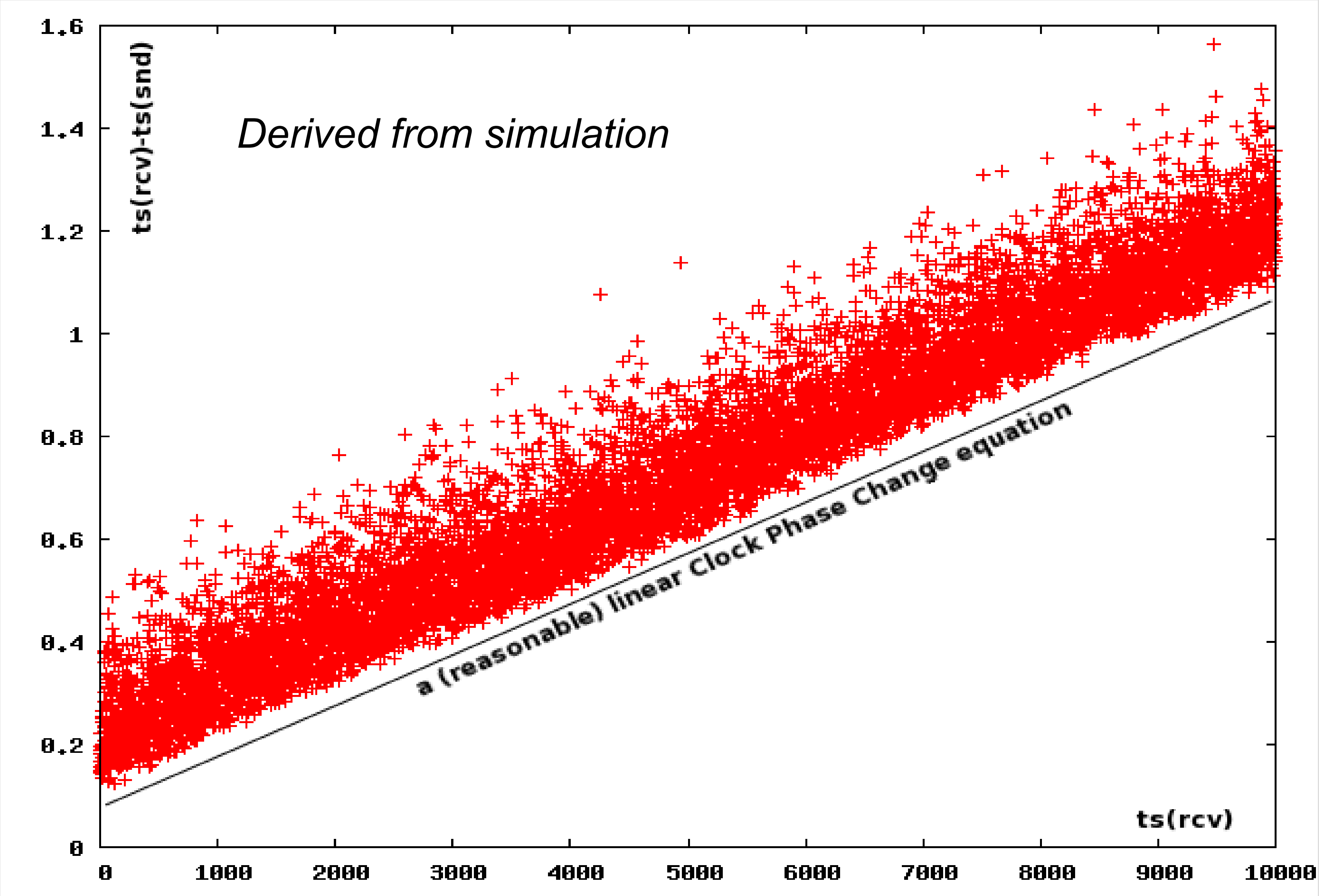
ts() - message send and receive timestamps, based on local clock;

Δ_i - real message latency;

a - a linear component of the Clock Phase Change;

b - a constant component of the Clock Phase Change.

Timestamp difference plot

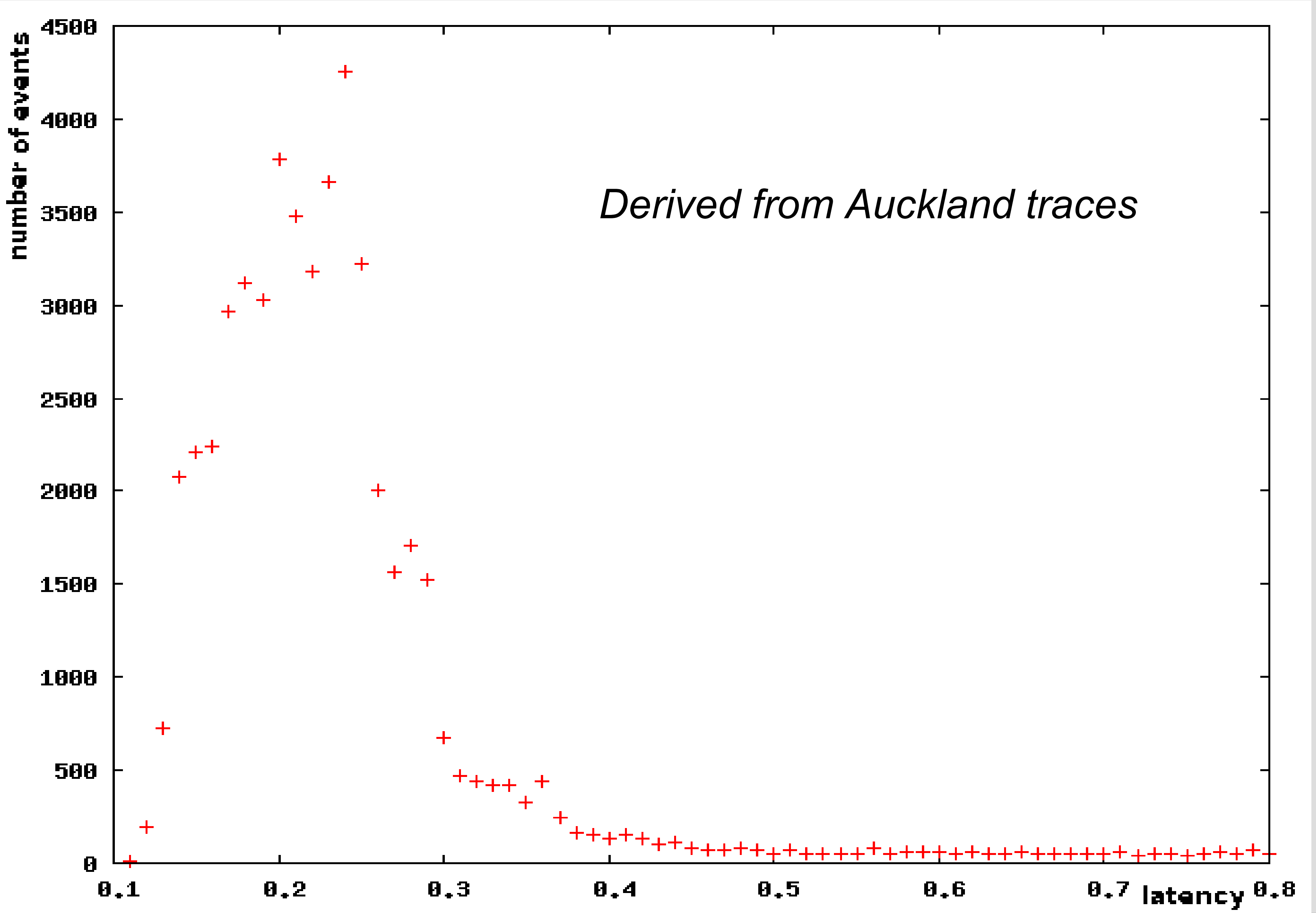


Clock Phase Change Interpolation

$$ts(rcv_i) - ts(snd_i) = \Delta_i + a * ts(snd_i) + b$$

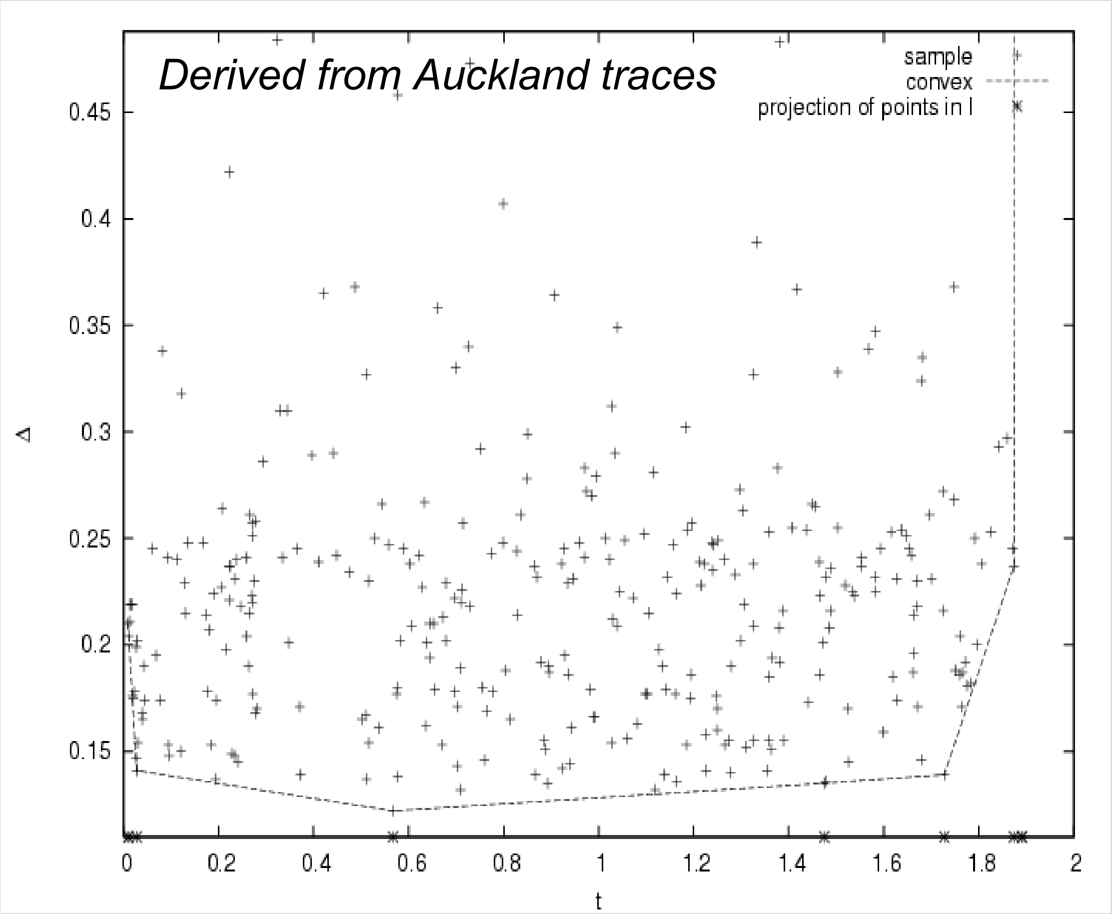
- To compute a constant term of Clock Phase Change, two values for index *i* are required, such that two real message latencies are identical.
- Based on experience two minimal values of message latency in a sample are likely close.

Message latency distribution



Finding the minimals

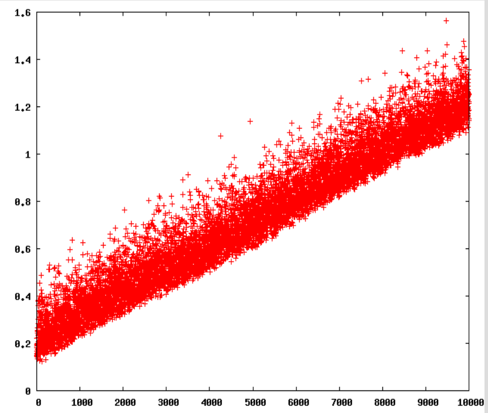
Points with minimal delay necessarily correspond to adjacent vertices of the (lower) hull containing a *timestamp difference* graph.



Proof: classical, by absurd.

The Graham Scan

```
@hull->empty;
while <($snd,$rcv)> {
    $new=($snd,($rcv-$snd));
    while (test($hull(N),$hull(N-1),$new)){ pop @hull }
    push $new,@hull;
}
```



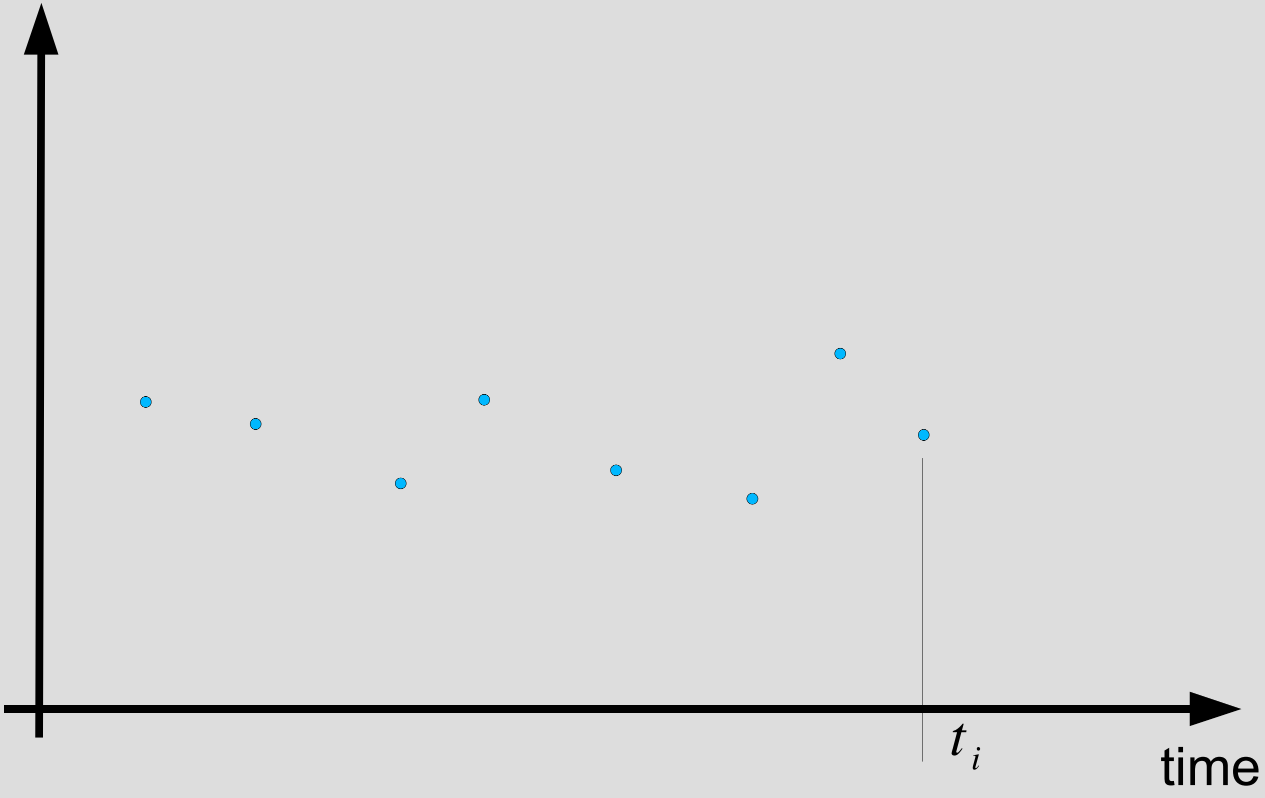
- the test function `test` computes and compares the slopes of the segments (`$hull(N-1),$hull(N)`) and (`$hull(N-1),($snd,$rcv`));

- the number of elements in `$hull` grows logarithmically with time.

Summarizing cost per time unit grows logarithmically with time.

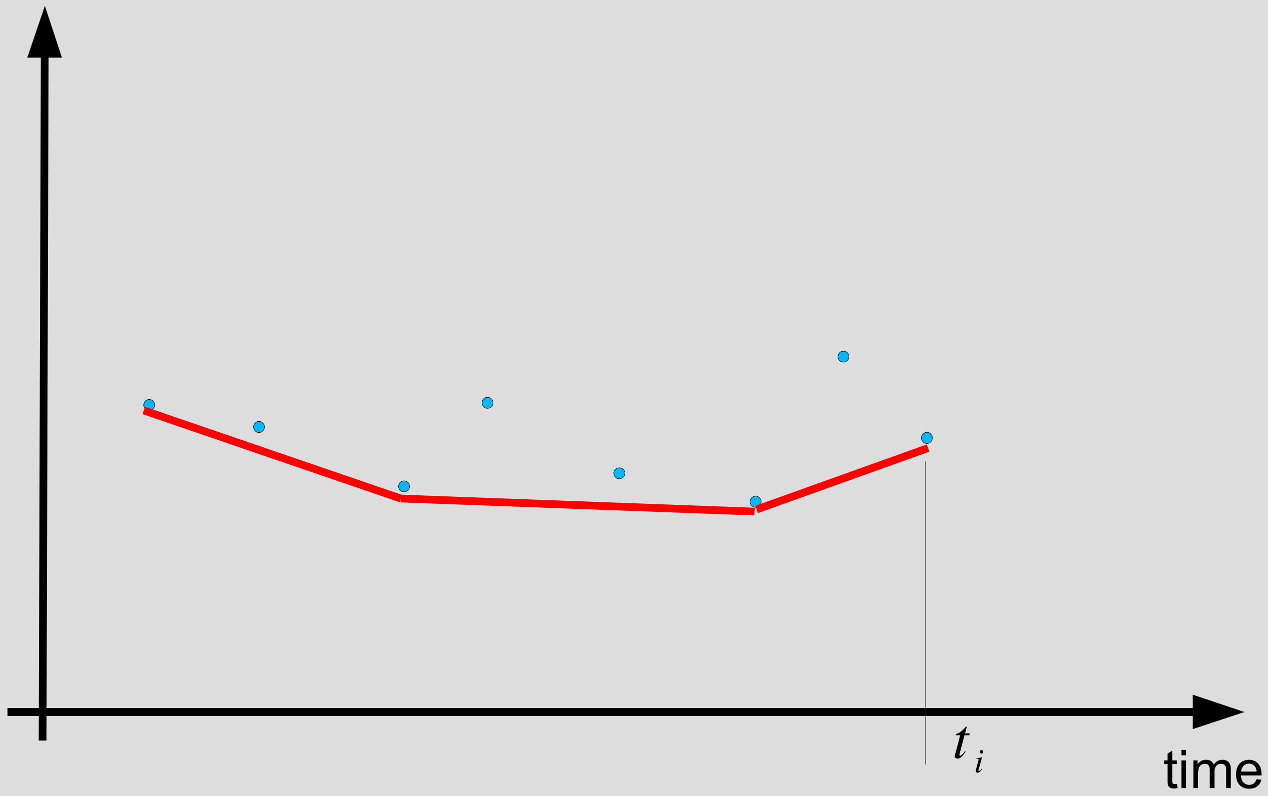
Graham Scan

A set of points, representing timestamp differences



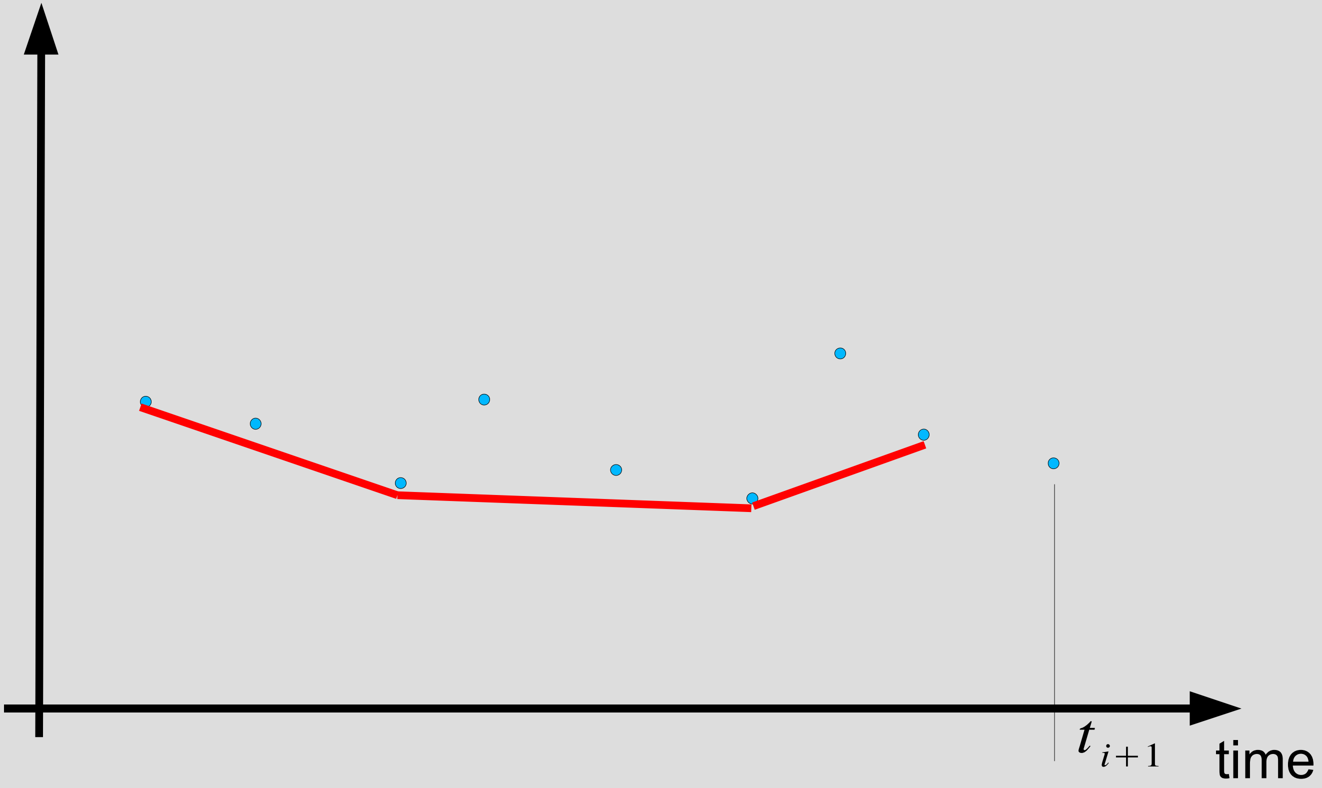
Graham Scan

The convex hull



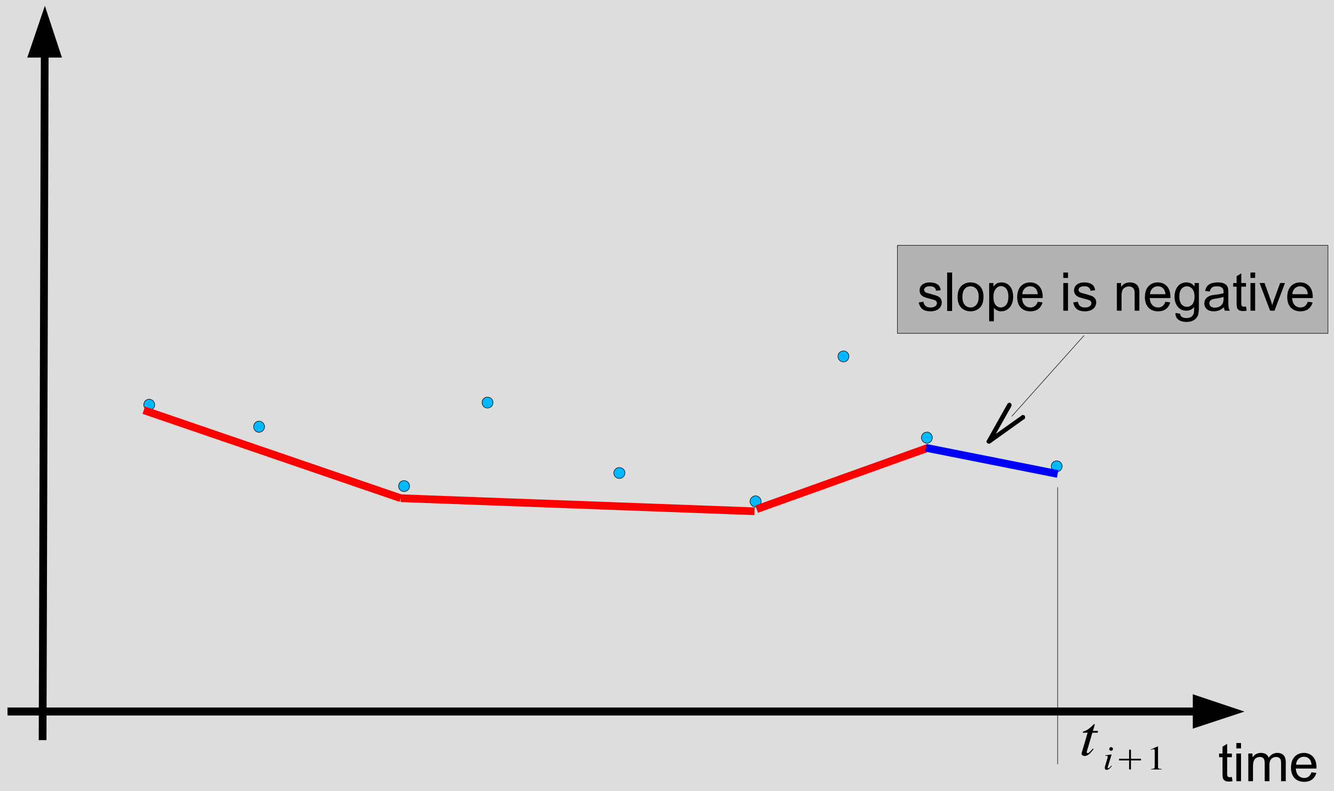
Graham Scan

A new point is added to the set



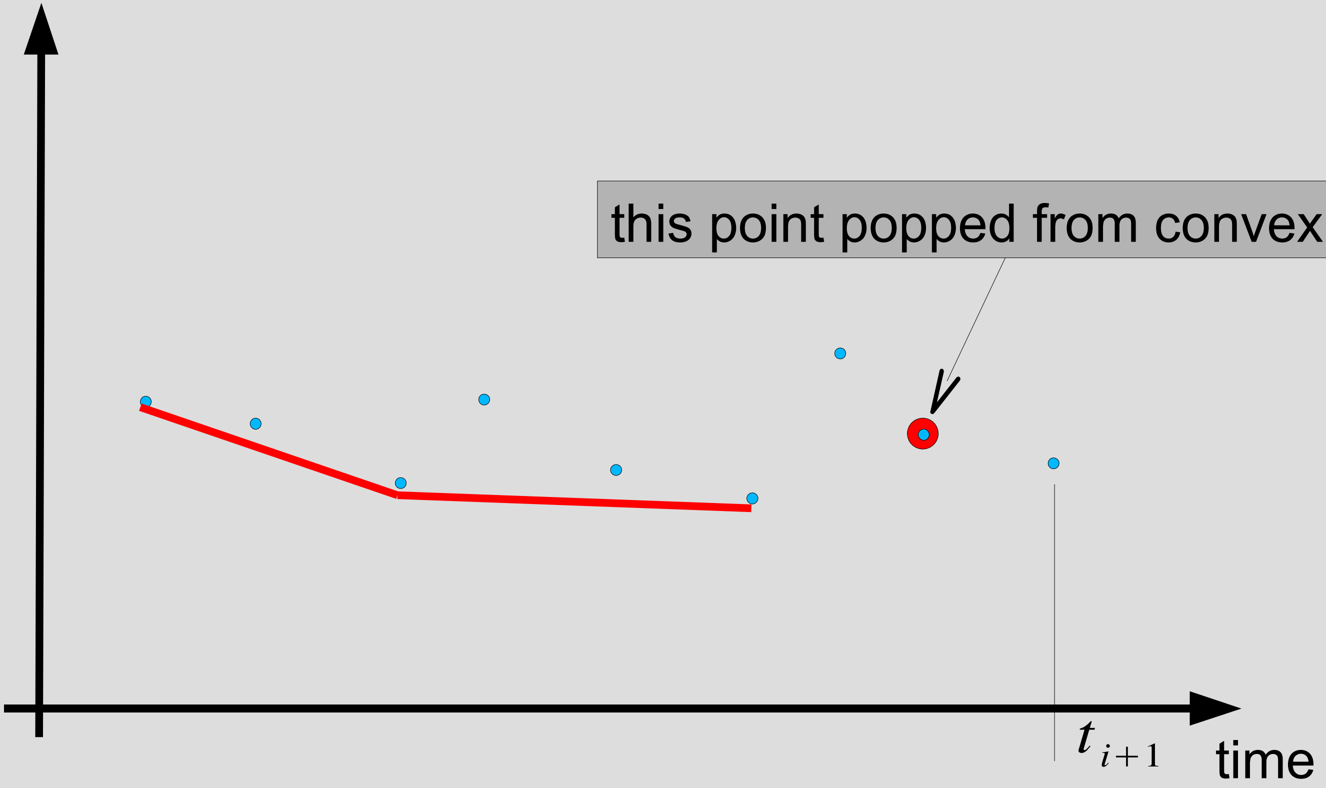
Graham Scan

The slope to the last point in the hull is computed



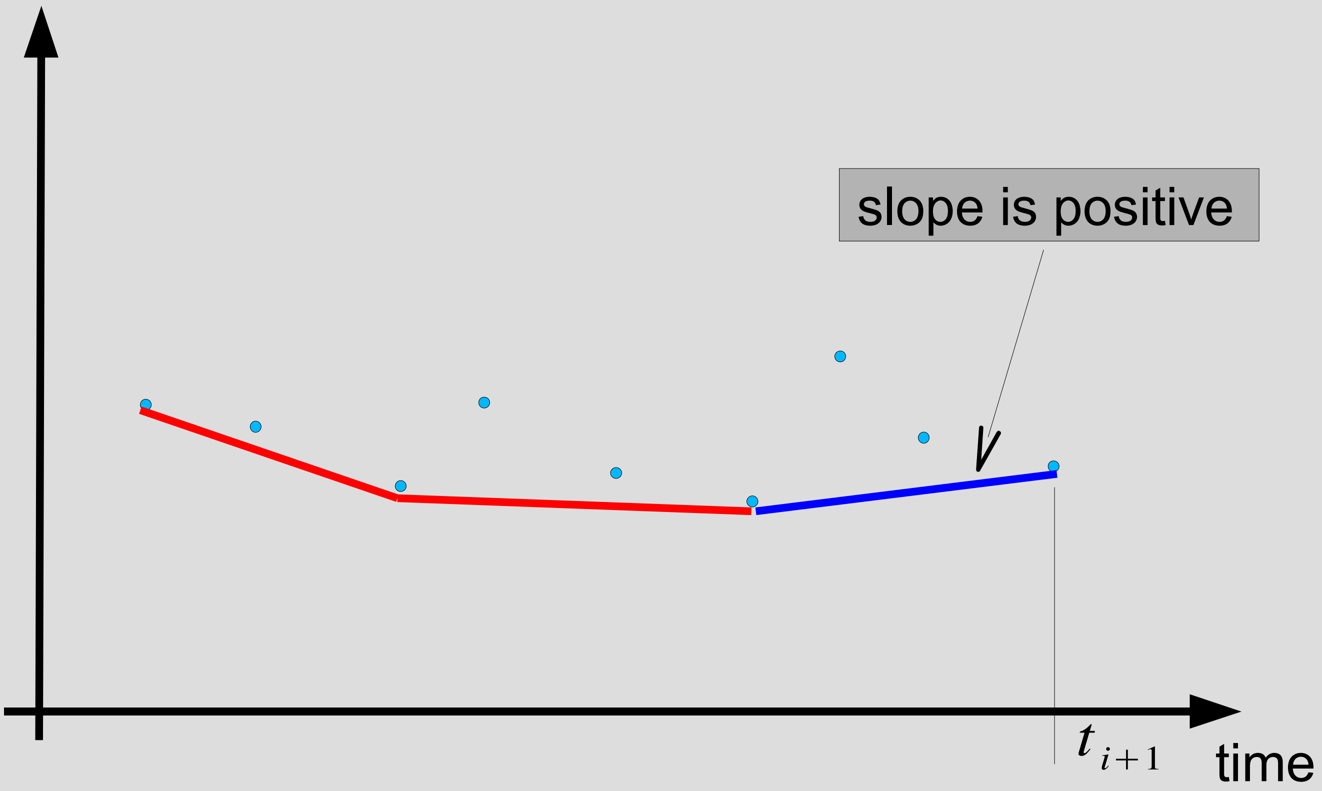
Graham Scan

The point is eliminated from the convex hull



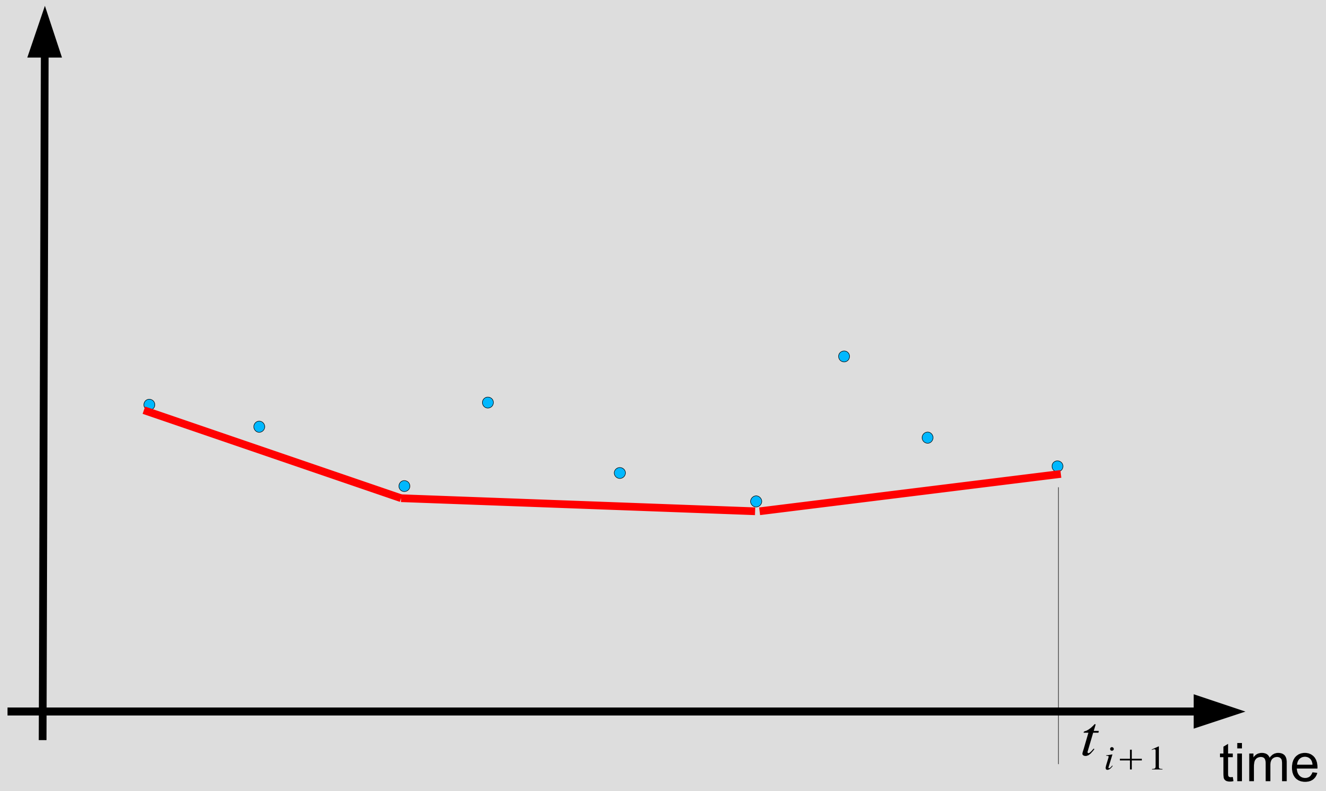
Graham Scan

Compute slope to next point



Graham Scan

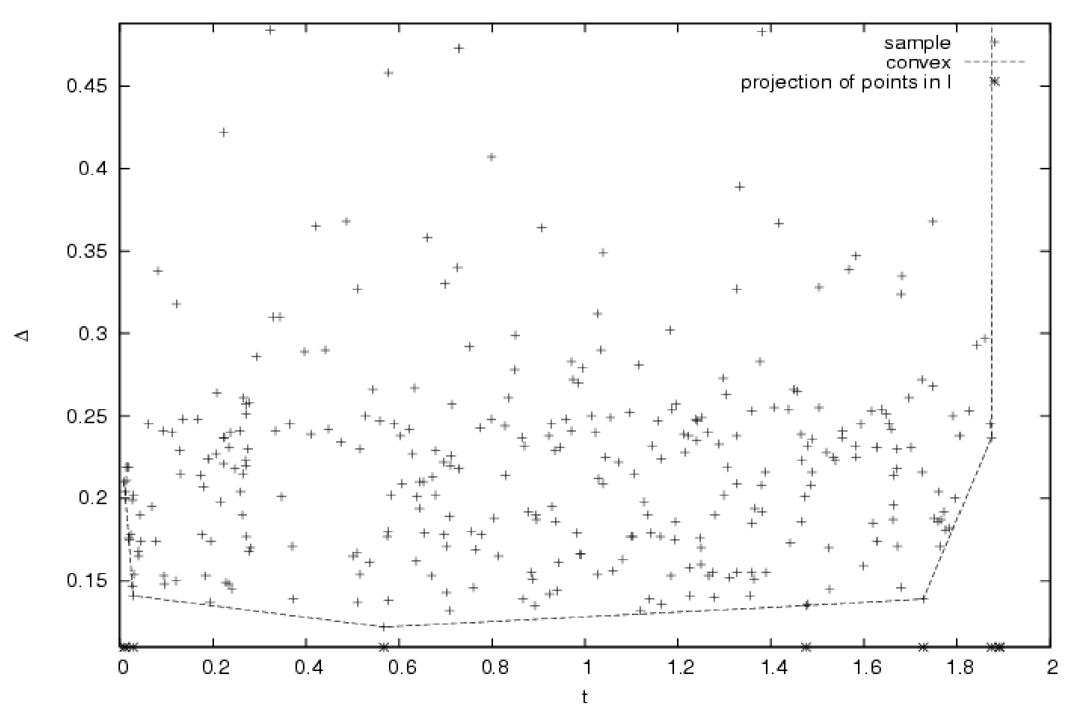
New point is pushed in the stack; exit.



Selecting the minimals

Selection rule:

select edges with farther sampling time in `$hull`



- rationale:
  - minimizes worst case error of the estimate
  - easy to compute
- more investigation needed

Evaluating estimate accuracy

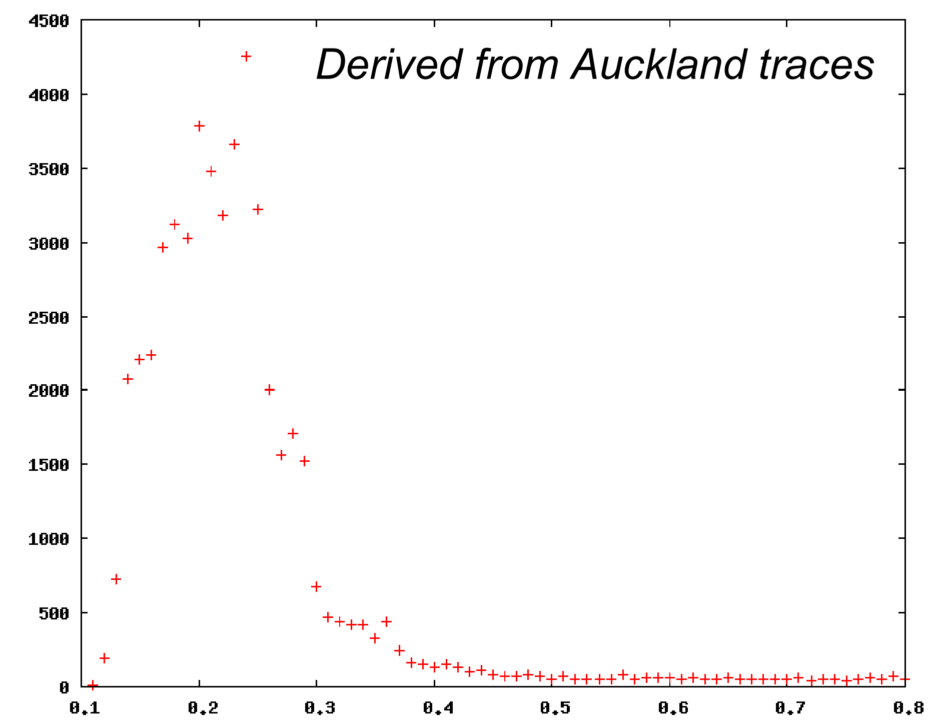
- Depends on communication delay distribution
- Increases with sample size
- May be deceived by relevant non-linear (temperature driven) Clock Phase Changes
- May be deceived by interfering clock adjustments

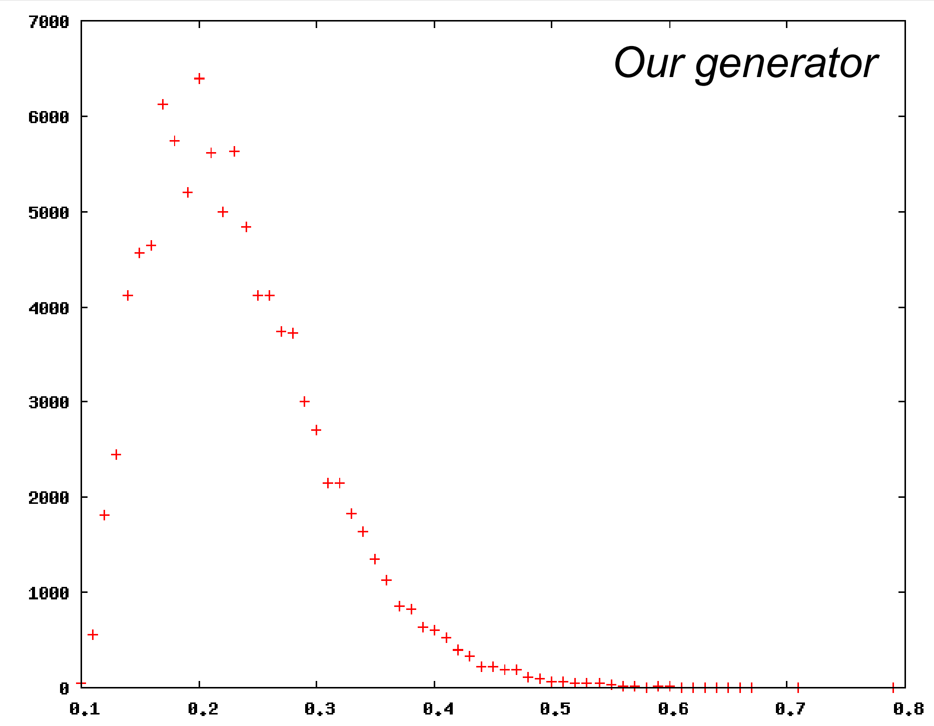
A simulation enlightens long run aspects of the Clock Phase Change estimation algorithm.

Simulation basics

A key is our generator of round-trip delays.

Our generator is fast and simulates long range dependence.



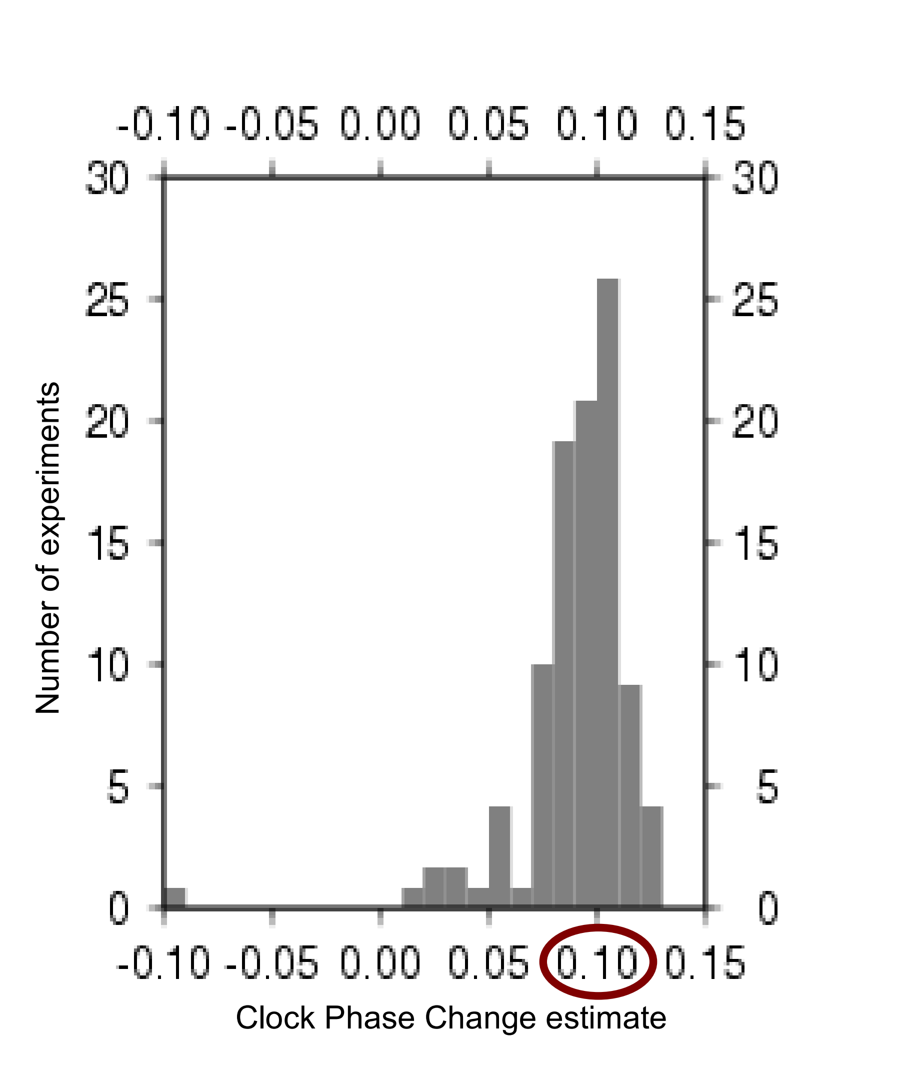


It is tuned on Auckland samples and introduces thermal variations.

Results: accuracy after stabilization

120 samples are generated with a Clock Phase Change of 0.1 parts per thousand (100 times better than a quartz clock) with periodic thermal shift.

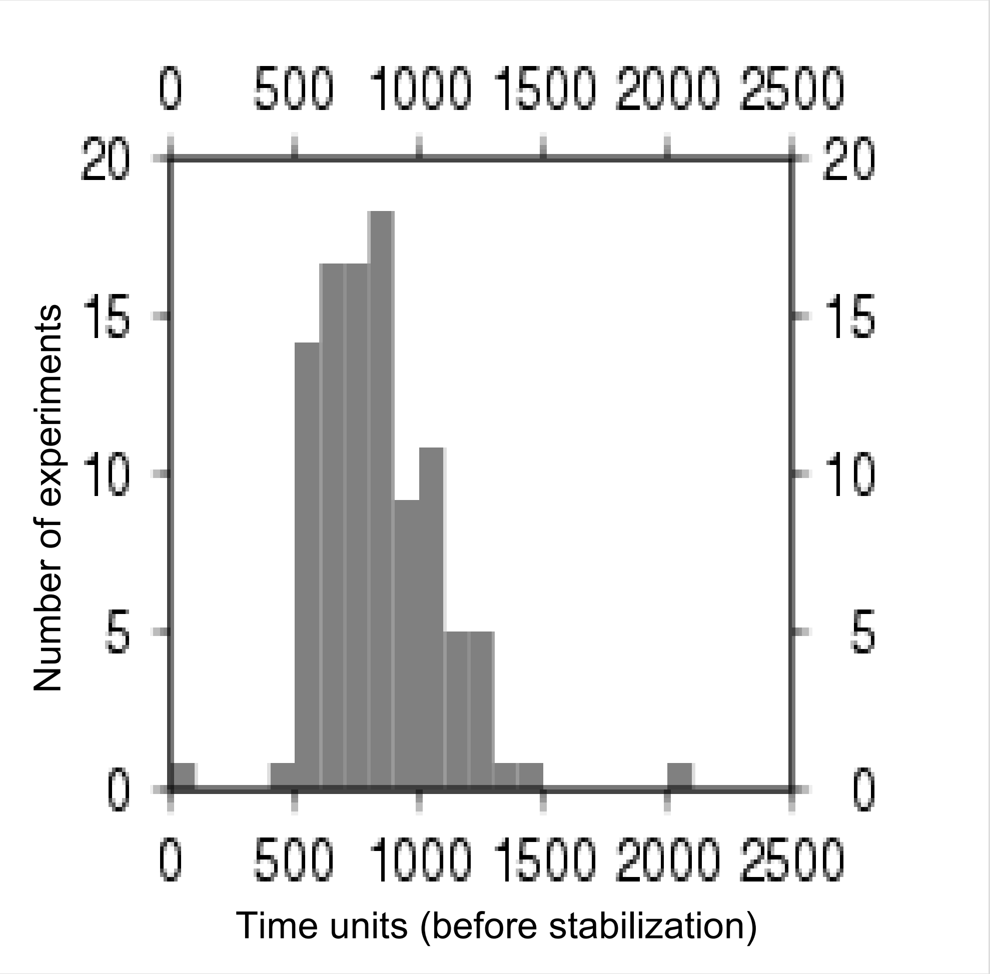
Estimate is read the first time the algorithm stabilizes (small variation of successive estimates).



Results: time to converge

Stabilization occurs when variation of successive estimates is small.

At 1 ping per second stabilization is reached after 20 minutes.



Conclusions

- Clock Phase Change compensation improves performance of clock synchronization.
- An efficient algorithm may significantly reduce Master to Slave updates' frequency, while maintaining required timing accuracy.
- Graham Scan algorithm offers an efficient low cost/impact solution.

Conclusions (continued)

- Clock Phase Chance compensation using Graham Scan brings savings in
  - Cost (by using cheaper oscillators),
  - Utilized bandwidth (by less frequent messages) and
  - Power consumption (by using one-way messages).
- Temperature/ageing component remains to be compensated.

References

- Moon, Skelley, Towsley “Estimation and Removal of Clock Skew from Network Delay Measurements”, TR98-43, Univ. of Massachusetts at Amherst.
- Cristian “Probabilistic Clock Synchronization”, Distributed Computing, 1989
- de Berg, van Kreveld, Overmars, Schwarzkopf *Computational Geometry*, pag 1-8, Springer 98.
- http://search.cpan.org/~augusto/Time-Skew-0.1/Skew.pm

Design of a One Way Jitter estimator

- One way delay (from previous formula) is:

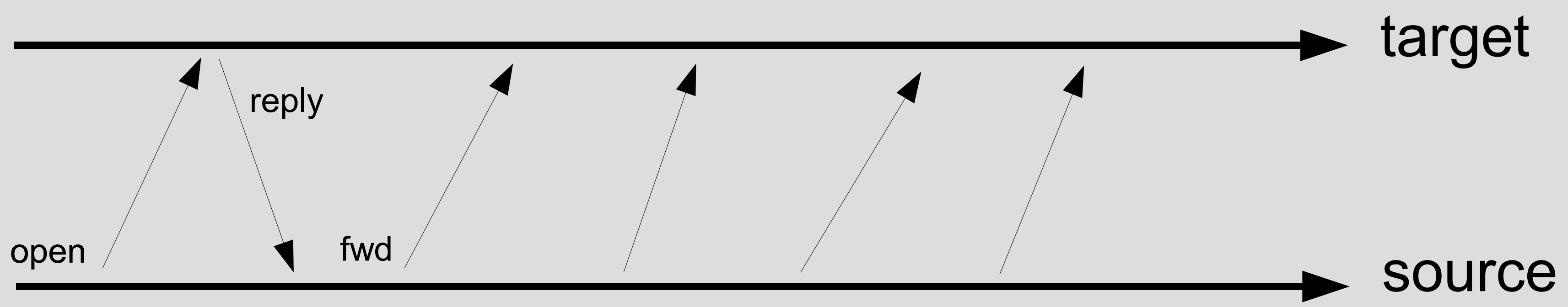
$$\Delta_i = ts(rcv_i) - ts(snd_i) - (a * ts(snd_i) + b)$$

As a consequence:

$$\Delta_i - \Delta_{i-1} = ts(rcv_i) - ts(rcv_{i-1}) - (1+a)(ts(snd_i) - ts(snd_{i-1}))$$

Therefore we conclude that only Clock Phase Change is needed to compute one-way drift.

A tool for OW jitter measurement



The protocol centers around three types of packets:

- open: request of authorization to ping;
- reply: contains authorization and parameters;
- fwd: measurement messages (timestamped).

JaMeter: a prototype

JaMeter is a monitoring tool that measures OW jitter both forward and backward.

- originally designed to measure asymmetry in the jitter (JA stands for jitter asymmetry);
- can produce results either on a dedicated MySql database, or on the stdout;
- currently deployed as part of GlueDomains