Distributed Computing and Systems
Chalmers university of technology

Competitive Freshness Algorithms for Wait-free Objects

Peter Damaschke, *Phuong Ha* & Philippas Tsigas

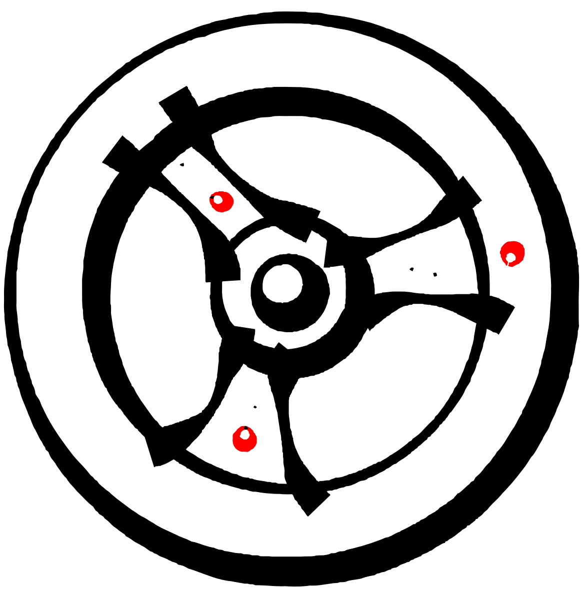
Presentation at the Euro-Par 2006
29th Aug. – 1st Sept. 2006, Dresden, Germany.

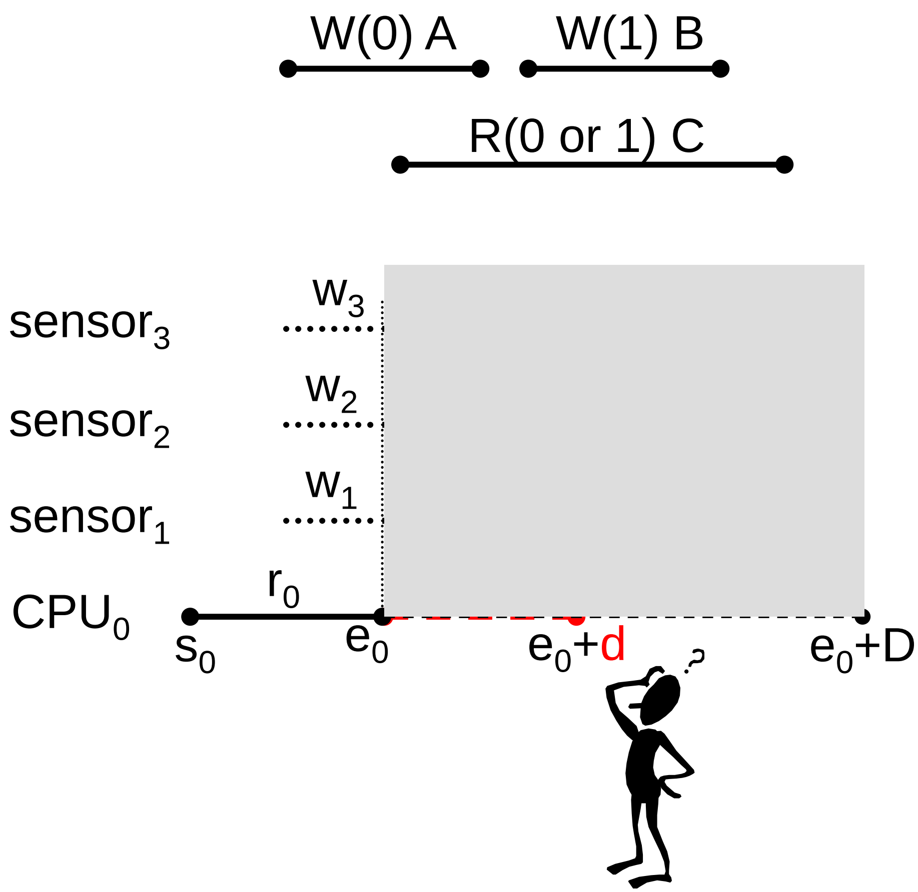
Wait-free data objects

- Concurrent data objects
  - Consistency!
- Solutions:
  - Mutual exclusion?
    - ⇒ risks of lock-convoy, deadlock & priority inversion ☹
  - Non-blocking synchronization
    - Wait-free:
      - every operation is guaranteed to finish in a limited number of steps.
      - ⇒ Suitable for real-time systems

Freshness

- Reactive systems need read-operations that both *respond fast* and *return fresh values*





Earlier work

- Freshness in databases
- Freshness in caching systems
- Freshness for concurrent data objects
  - single-writer-to-single-reader asynch. comm.

Contributions

- The first paper that attacks the freshness for multi-writer multi-reader shared objects
- Competitive freshness
  - An optimal deterministic algorithm
  - A nearly-optimal randomized algorithm

Road-map

- Introduction
- Modeling the problem
- Optimal deterministic algorithm
- Nearly-optimal randomized algorithm
- Conclusions

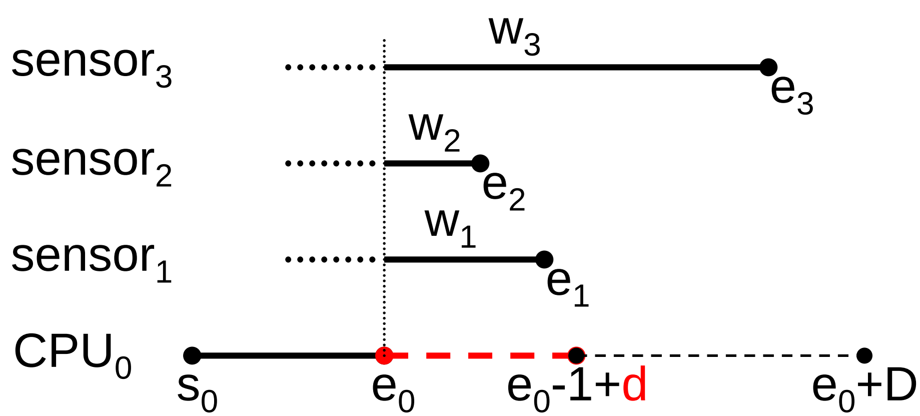
Model

- Assumptions:
  - An upper bound *D* on operation execution time

- Freshness $f_d = \frac{|e_d|}{d}$

- Constraints $\frac{|e_{d-1}|}{d} \leq f_d \leq \frac{M}{d}$ and $\frac{M}{D} \leq f_d \leq M$

d: delay, $1 \leq d \leq D+1$
$|e_d|$: # fresh values
M: # concurrent writes at e_0



Freshness as an online game

Freshness

$$\ln M - \ln D \le \ln f_d \le \ln M$$

$$\ln|e_{d-1}| - \ln d \le \ln f_d \le \ln D - \ln d$$

Online game:

- player (read operation)

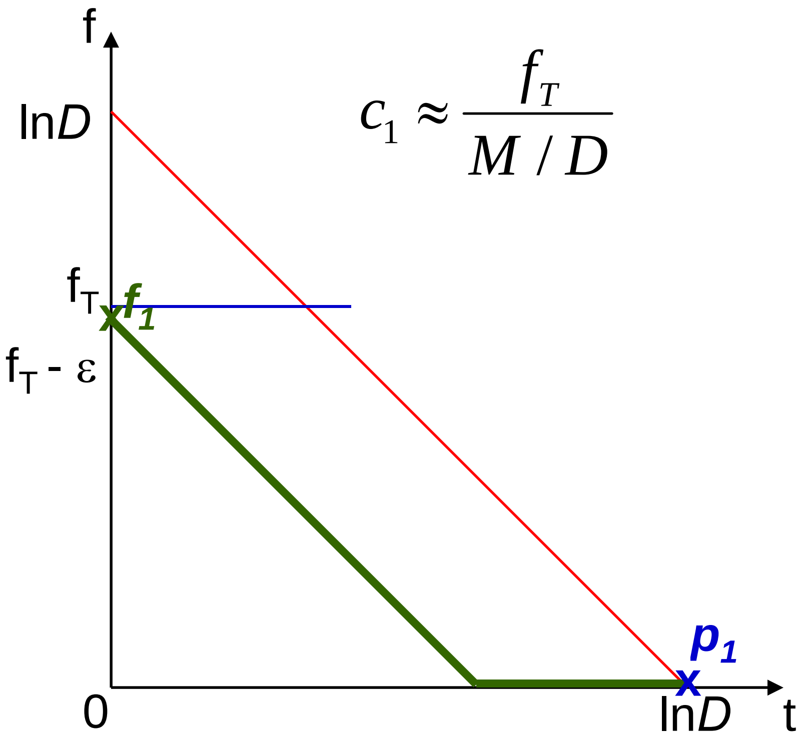
  vs.

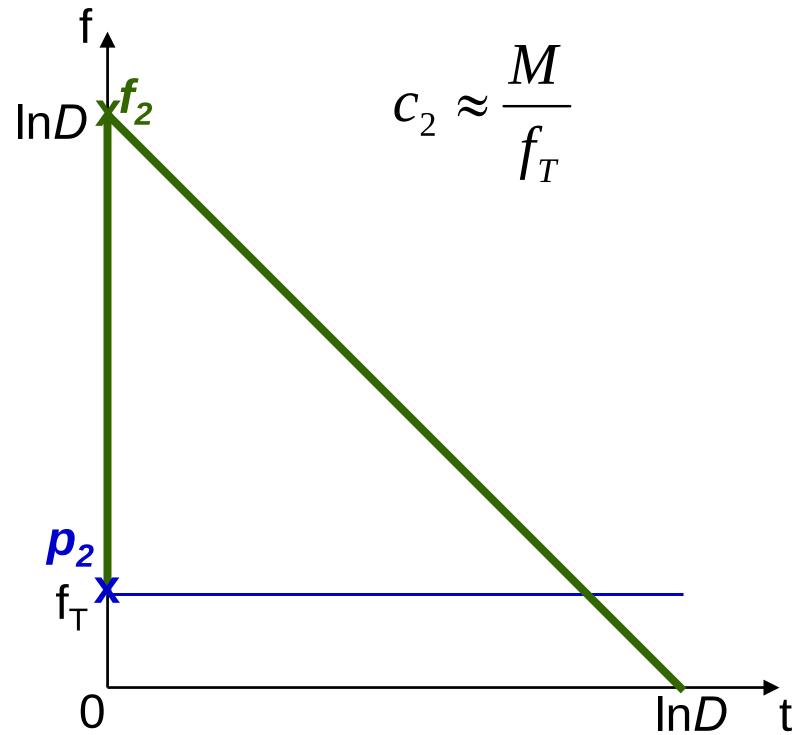
- malicious adversary

A deterministic algorithm

Algorithm: The read accepts the first $f_d \geq \frac{M}{\sqrt{D}}$

Analysis:





$$\Rightarrow c_1 = c_2 = \sqrt{D}$$

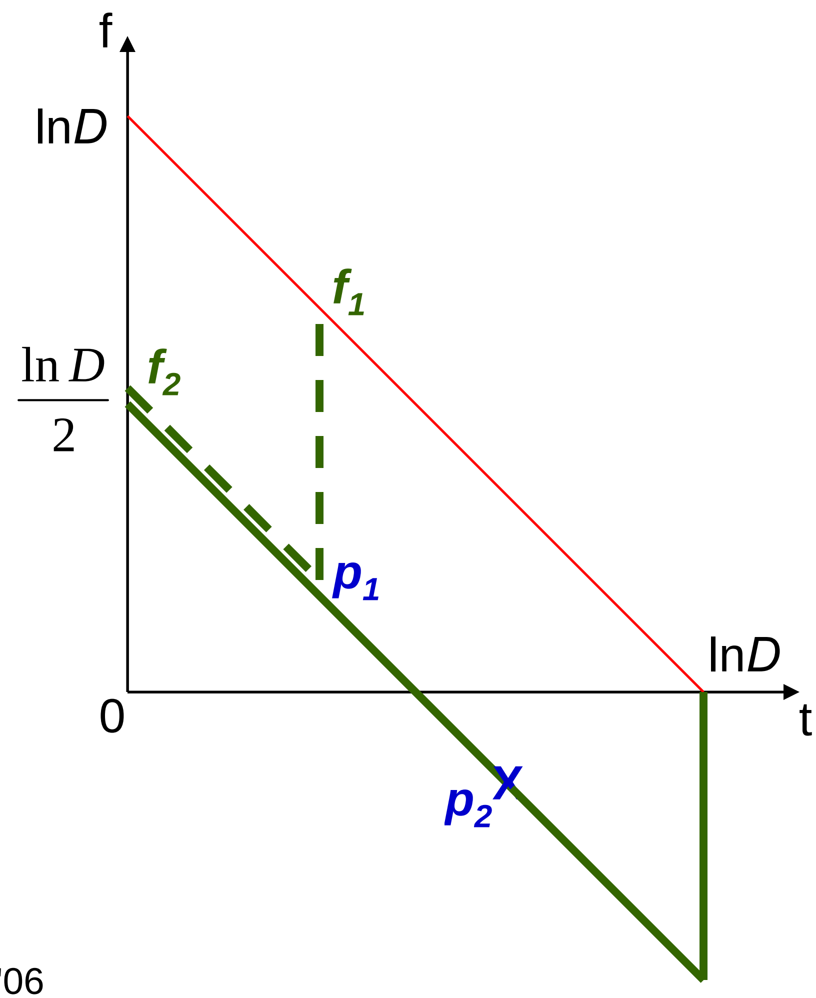
Lower bound $\sqrt{D}$

- Adversary's strategy:
  - Start with $c = \ln(D)/2$ & decrease c at unit speed until the player stops. At this time,
    - if $c > 0$, c jumps to the max.
    - if $c \leq 0$, c keeps decreasing
- Case 1

  $f_1 - p_1 = \ln D / 2$
- Case 2:

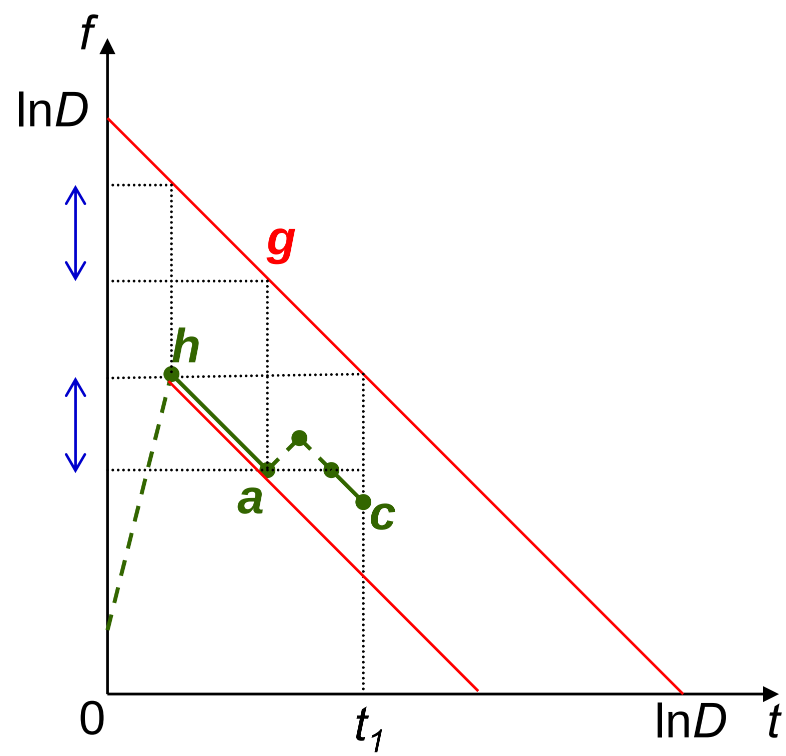
  $f_2 - p_2 \geq \ln D / 2$

$\Rightarrow$ Comp. ratio = $e^{(f-p)} \geq \sqrt{D}$



A randomized algorithm

- Ideas:
  - Put a probability on the freshness *c* when it starts to go down.
- Algorithm
  - When *c* is decreasing, put on it a probability $p = 2 / \ln D$
  - If the game is over (i.e. *h*=*g*), put the rest *r* on the current *c*



A randomized algorithm

- Competitive ratio

$$c = \frac{\ln D}{1 + \ln 2 - \frac{2}{\sqrt{D}}} \xrightarrow{D \to \infty} \frac{\ln D}{1 + \ln 2}$$

- Optimal randomized comp. ratio *(*ln *D)/2,* asymptotically
  (cf. TR-CS-2005:17)

Conclusions

- The first paper that defines the freshness problem for wait-free data objects.

- Competitive freshness
  - An optimal deterministic algorithm
  - A nearly-optimal randomized algorithm

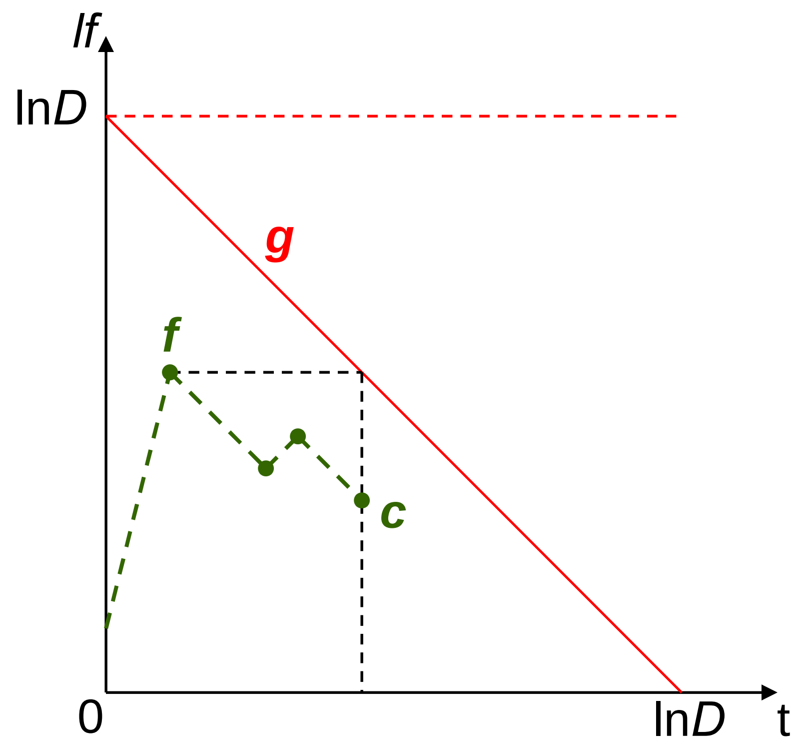
- Contributions to the online search problem
  - New general models

Thank you for your attention!

A randomized algorithm

- Comp. ratio $c = \dfrac{\ln D}{1 + \ln 2 - \dfrac{2}{\sqrt{D}}}$
- Randomized online search $\rightarrow$ deterministic one-way trading:
  - exchanging some fraction of money $\approx$ stopping the search with that probability
- Conventions:
  - distributed money on axis *lf*
  - *T(x)*: density of exchanged money

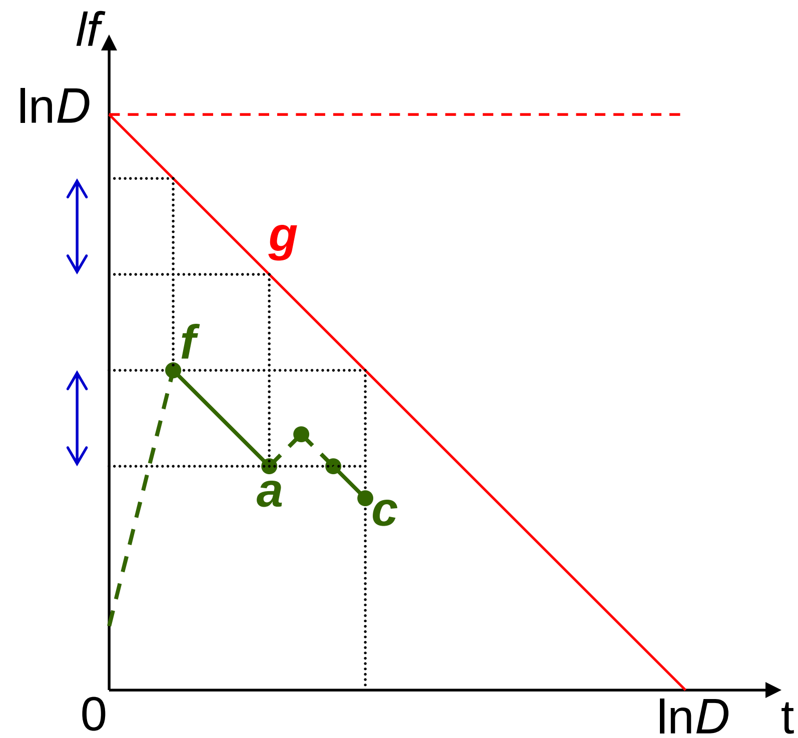
$\Rightarrow$ player's profit = $\int (x.T(x))$



Randomized algorithm

- Ideas:
  - Exchange //////// when *c* starts to go down.

comp. ratio



- Optimal comp. ratio *O(ln D)*

(cf. TR-CS-2005:17)

Analysis

- Let x = f – c (final value)
- Observations:
  - *T=2* on *[c,f]* or *c=f*
  - $\sum$ *(gaps with T=0)* $\leq r$
- Player's profit

$$\geq f.\min_{r,x}\left(2\int_0^x e^{-t}dt + re^{-t} + 2\int_{x+r}^{(r+\ln D)/2} e^{-t}dt\right)$$

$$> f\left(1+\ln 2 - \frac{2}{\sqrt{D}}\right)$$

- Adversary profit: *f.ln D*

$\Rightarrow$ comp. ratio $c = \dfrac{\ln D}{1+\ln 2 - 2/\sqrt{D}}$

- Optimal comp. ratio *O(ln D)*

(cf. TR-CS-2005:17)