Operational Replication Implies Real-Time

Gérard Le Lann

INRIA – Paris-Rocquencourt Center, France

- **Pedagogical (definitions)**
- **Sociological (we have had limited impact on reality)**

Database (DB) community:

To increase performance, diversification rarely considered

Distributed computing (DC) community:

To achieve fault tolerance, diversification considered

Replication

Abstractions (system models, computational models, fault models, fault occurrence models) and algorithms proved to meet performance and/or fault tolerance requirements.

Operational replication

Replication proved to meet performance and/or fault tolerance requirements stated for real/deployed systems & applications, i.e. instantiated replication.

Why operational replication?

Michael J. Fisher, Michael Merritt, 2003, in “Appraising Two Decades of Distributed Computing Theory Research”:

“It is difficult to apply theoretical results to practical situations, for one or more of the underlying assumptions will often fail to be satisfied”.

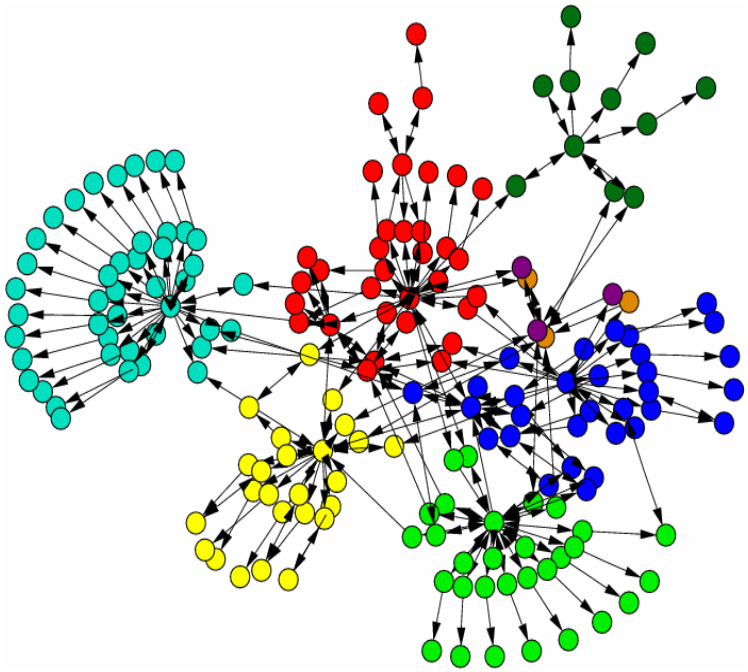
Fred Schneider’s talk: “Assumptions are potential vulnerabilities”.

Replication is not a panacea (especially if used inappropriately)

- The Ariane 5/flight 501 failure (1996)
- The 20-hour crash of Bouygues Mobile Tel. data center (2005)
- …/…

Real-Time

A (distributed) system multiplexes many high-/…/low-level processes, activated according to various event occurrence models. Inevitably, resource contention and waiting queue phenomena develop. Hence the need for schedulers driven by problem-dependent parameters.

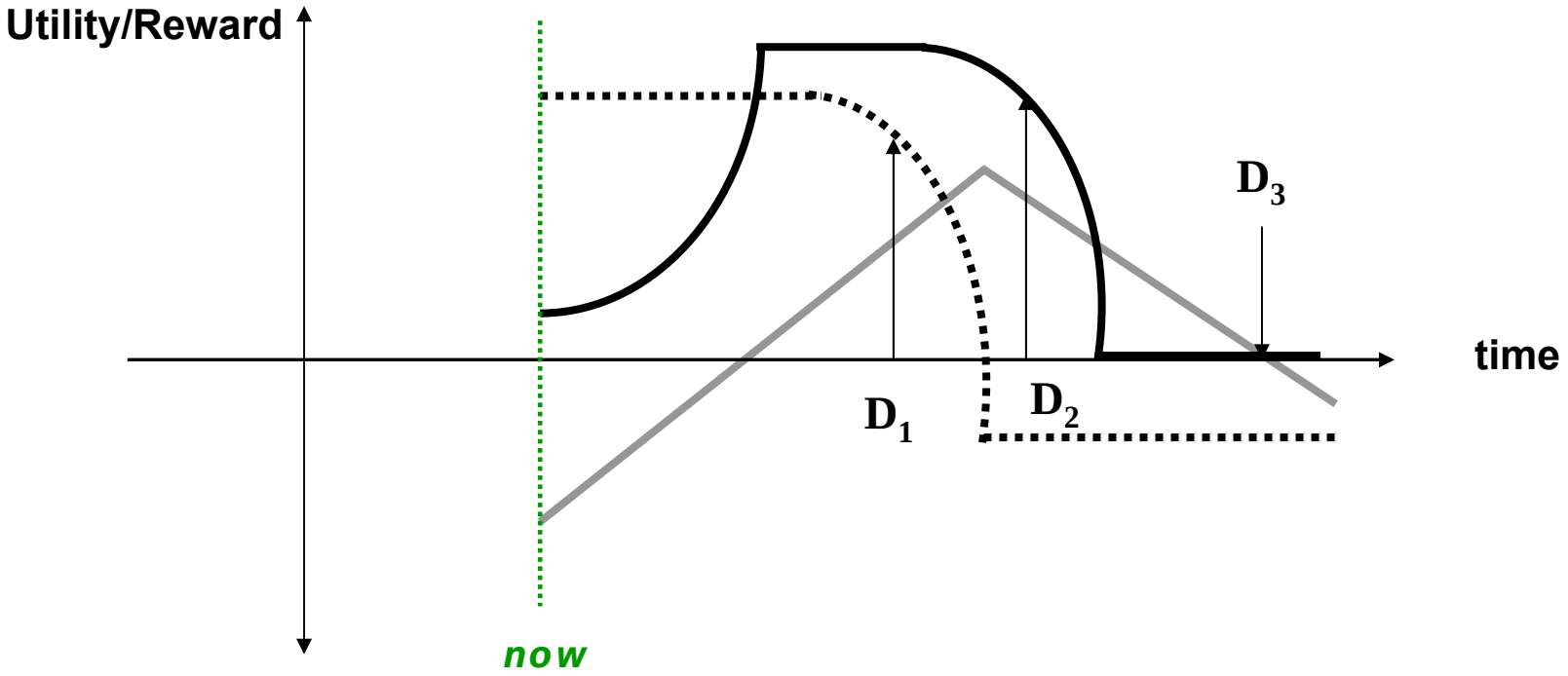


Real-time computing?
Problems explored in ***complexity theory, combinatorial optimization, game theory, scheduling theory, decision theory***, …

The essential difficulty with real-time computing problems:
Expressing tight upper bounds for sojourn times in waiting queues, given some couple {adversary, scheduler}

A real-time computing problem is solved iff feasibility conditions (timeliness properties, maximization of reward functions, …) are established—via analytical techniques, in appropriate algebrae (e.g., (max,+) algebra),…

➔ timeliness properties



At time *now*, a scheduler must "find out" an ordering of execution for the 3 pending processes that meets all deadlines D_i if feasible, or that maximizes the aggregated reward if at least one deadline cannot be met.

- With TimeP:
  - Analytical upper bounds on response times – B(k) for process k – are established for every process
  - D(k) – k's strict termination deadline – is given (problem specification)

$$\Rightarrow \quad \forall\, k, \quad B(k) < D(k)$$

- With TU-TimeP:
  - Analytical lower bounds of achieved utilities – U(k) for process k – are established for every process
  - V(k) – k's highest utility – is given (k's TUF in problem specification)
  - $0 < \alpha < 1$, T stands for any given time interval, K(T) = set of processes that may terminate within interval T

$$\Rightarrow \quad \forall\, T, \forall\, k \in K(T), \quad \sum_T U(k) / \sum_T V(k) > \alpha$$

Response times – Throughput in DB or DC systems

- End-to-end/peer-to-peer/client-server ***throughput*** **(THR)**

  THR: Effective throughput (in the presence of adversary activity, i.e. impacted by sojourn times in waiting queues)

  $$THR = 1/d$$

  d: effective inter-symbol delay, i.e. delay between two consecutive steps (step: transmission of a byte, computation of state transition)

- End-to-end/peer-to-peer/client-server ***response times*** **(R)**

  $$R = \Delta + Inf/THR$$

  Δ: Propagation delay (includes sojourn times in waiting queues)

  Inf: Quantity of information processed and transmitted

DB: Replication for performance

Properties of interest: response times, throughput

- R and replication? Copy locality reduces Δ ➔ reduces R
- THR and replication? Copy locality minimizes number of waiting queues visited ➔ reduces d ➔ augments THR ➔ reduces R

Reduction of bounds on response times and inter-symbol delays implies knowing how bounds depend on replication degree, copy localizations, … ➔ Implies bounds are computable ➔ Implies real-time. Average bounds: soft real-time. Upper bounds: hard real-time.

Replication for the DB community implies real-time

A fortiori…

Operational replication for the DB community implies real-time

DC: Replication for fault tolerance

Properties of interest: (logical) safety, liveness (termination)

- (Logical) safety and replication? Unbounded or infinite silence is possible.
- Liveness and replication? Upper bounds on response times or inter-symbol delays are unknown.
- Termination and replication? Upper bounds on response times or inter-symbol delays are (said to be) known—assumptions.

Replication for the DC community?

Worst-case lower bounds for (logical) time complexity

No concern for upper bounds on times/delays, except for Termination

Prima facie…

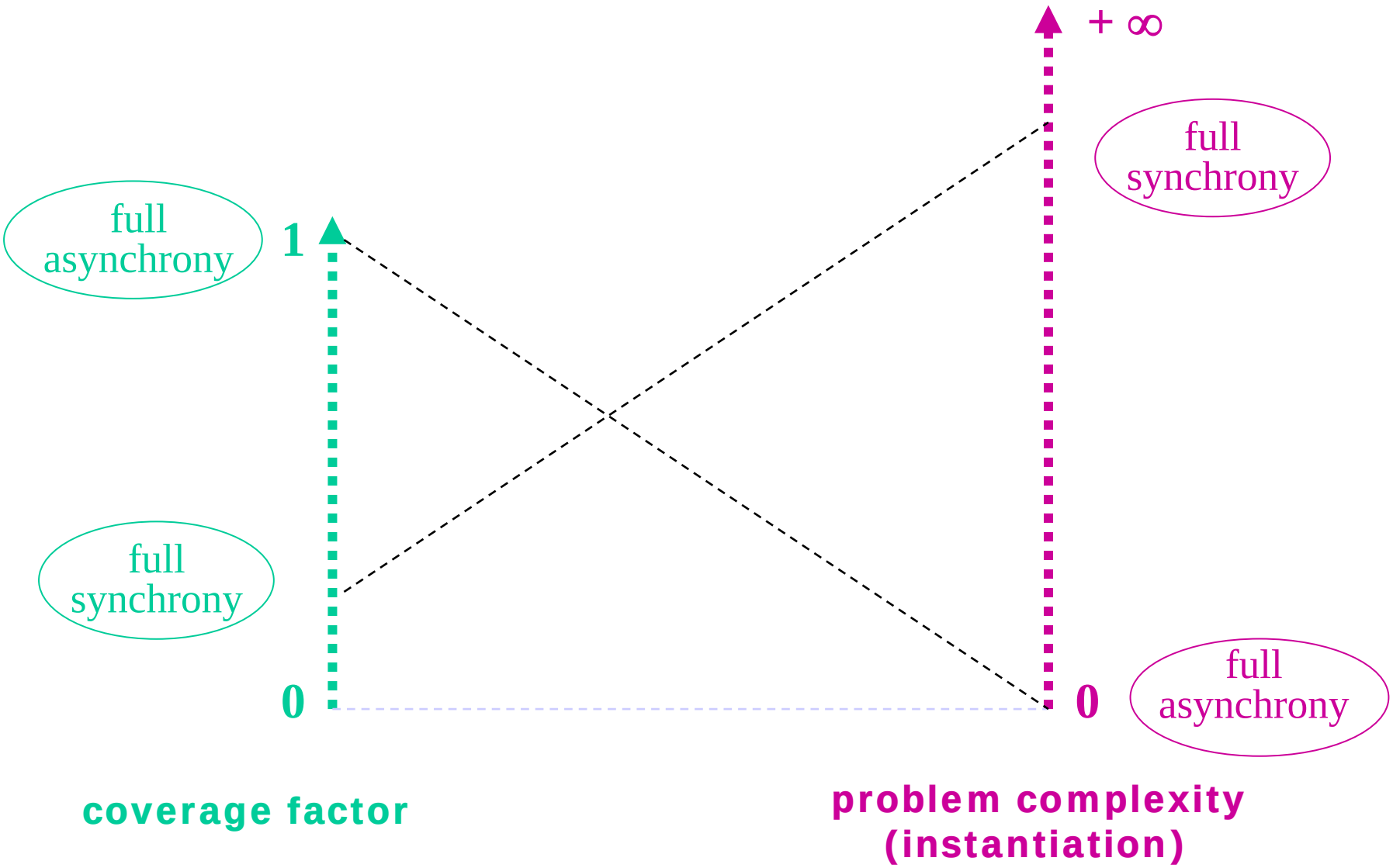
Replication for the DC community does not imply real-time

DC: Operational replication for fault tolerance?

Implies showing how to instantiate abstractions under which proofs of required properties hold true (with a given algorithm):

- system models,
- fault models,
- fault occurrence model(s),
- computational models.

System models



System models

Except with full asynchrony, we assume “something” regarding time/delay upper bounds. Examples:

- Communication-close rounds,
- “Not too many” violations of postulated bounds (timing faults),
- Full synchrony assumed for some sub-system (e.g., timely links),
- …/…

How “trustable” these assumptions, in a given real universe/system, i.e. where a given set of processes might be executed in contention-prone ways (not just your favorite Consensus algorithm)?

Impossible to know, unless analytical expressions of time/delay bounds are established.

Fault occurrence models

Inevitably, we assume “something” regarding fault occurrence density. Best example is the “up-to-f” assumption (any fault model).

How “trustable” such assumptions in a given real universe/system, i.e. where a given set of processes might be executed in contention-prone ways (not just your favorite NB Atomic Commit algorithm A)?

Example of an “embarrassing” question: “How do you know the value of f when you don’t know (1) the actual execution time of a round, (2) how long it will take for A to run to completion, in the presence of preemption?”

Intuitively, if f’ is postulated for a contention-free run, real f can only be greater than f’. How much f?

Impossible to know, with a coverage strictly equal to the coverage of f’, unless analytical expressions of time/delay bounds are established.

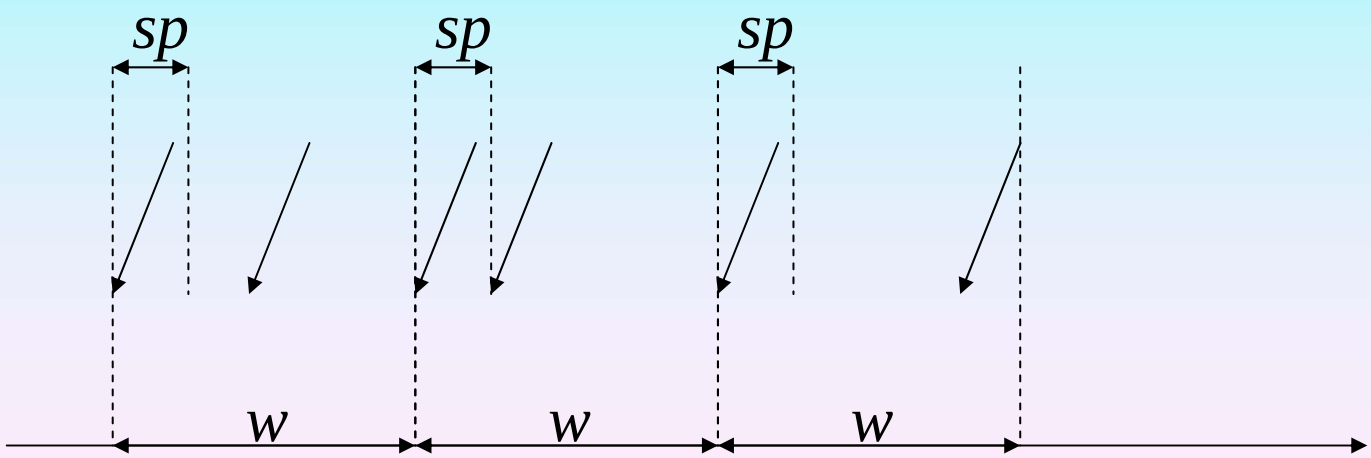
Problem is even more acute with transient/moving fault models!

Unimodal arbitrary (*uarb*), defined as triple {*w*, *a* , *sp*}:

***w* = sliding time window**

***a* = max number of arrivals within w**

***sp* = sporadicity interval within w**



Computational models

(1) Example of full asynchrony & Chandra/Toueg FDs

How “trustable” assumptions regarding completeness and accuracy, no FD-message losses, in a given real universe/system, i.e. where a given set of processes might be executed and might generate messages in various ways (not just the FD constructs)?

Impossible to know, unless analytical expressions of time/delay bounds for FD-messages are established ➔ worst-case schedulability analyses.

It is doable. Example: Fast FDs in IEEE Trans. on Computers, Aug. 2002.

Computational models

(2) Example of alternated “good” and “bad” periods

How “trustable” is it to assume such behaviors, in a given real universe/system, i.e. where histories of process executions are considered to be hardly predictable (under our models)?

More to the point, how can we tell whether “good” periods:
- occur “often enough”?
- last “long enough”?

How can we tell it’s past GST?

Impossible to know, unless analytical expressions of time/delay bounds are established.

Computational models

(3) Example of the HO model

The HO model eliminates inconsistencies that plague “traditional” models.
However, since predicates in the HO model encompass assumptions proper to system models and fault occurrence models, previous questions arise a fortiori.

Typically: How “trustable” is it to assume that some predicate holds, in some round?

Impossible to know, unless analytical expressions of time/delay bounds are established.

The Θ-model

Ratio Θ is *max (round duration)/min (round duration)*.

By definition:

max (round duration) depends on fault occurrences

max (round duration) comprises sojourn times in waiting queues

How “trustable” is it to assert:

- *max (round duration)* can be computed,
- Θ is not violated when *max (round duration)* is violated?

Give analytical expressions of time/delay bounds, for a given system model. It is doable. Example: Work by U. Schmid, J. Widder et al.

Conclusion

Operational replication for the DC community implies real-time

Michael J. Fisher, Michael Merritt, 2003, in “Appraising Two Decades of Distributed Computing Theory Research”:
“Real-time computing has historically been considered a distinct research area, but timing issues are too pervasive to be viewed as a subspecialty”.

Note that with DB systems also, it is mandatory to instantiate those abstract constructs under which proofs of required performance hold true:

- system models,
- fault models,
- fault occurrence models,
- computational models.

➔Additional reasons showing that, for the DB community, operational replication implies real-time

DC: Replication for fault tolerance?

…/…

Prima facie…

Replication for the DC community does not imply real-time

(concern here is of strictly theoretical essence)

Imagine that real-time computing problems “encapsulated” in some of our postulated abstract constructs are as “complex/hard” as those problems we solve in replicated DC …

From a strictly theoretical viewpoint, leaving them open is not satisfactory at all.

Fiction

The real-time computing community believes they solve distributed real-time computing/scheduling problems assuming, e.g., knowledge of global state, exact agreement in the presence of faults, atomic broadcast.

When asked about the nature of such assumptions, responses are:

Problems (if any) "encapsulated" in such assumptions are of little theoretical interest:

➔ ***to be solved by engineers,***

➔ ***they boil down to implementation issues.***

Who in the DC community would find this satisfactory?

Conclusion

Some real-time computing problems “encapsulated” in our postulated abstract constructs are as respectable as our distributed computing problems…

Replication for the DC community may imply real-time

Whenever the case, results from real-time/scheduling theory may be used to render your favorite DC algorithm “adversary-immune”.

Taking now a user viewpoint, distributed real-time fault tolerant applications are on the horizon (calling for “embedded systems”).

Many such applications/systems (where replication is commonplace) are safety-/life-/mission-critical: No timely responses (for the sake of logical safety) is no more acceptable than erroneous responses…

Conclusions: Operational Viewpoint

Need to consider problems involved altogether.

DB community

Be consistent with stated goals. Statistics collected a posteriori, or "fine tuning", … do not help in predicting performance, nor in identifying causes of poor performance. Conservative delay estimates lead to poor performance ➔ ***must solve real-time computing problems involved with operational replication.***

DC community

So far, we have not done it well enough (at all?). This is one of the causes for our limited impact on reality ➔ ***must solve real-time computing problems involved with operational replication.***

Conclusions: Theoretical Viewpoint

Need to consider all problems involved more carefully.

DB community and DC community

Granted, one cannot get rid of assumptions. However, whenever assumptions translate into problems that have solutions, we should not ignore them, nor picture them as "uninteresting issues".

Moreover, real/operational DB/DC systems/applications raise composite problems (e.g., security & reliability & distribution & real-time & safety & …) which are incredibly complex, barely explored by theoreticians.

Why aren't we working on these open composite problems?

That would imply defining adversary models that are more realistic than those contemplated so far (e.g., combined occurrences of transient faults, attacks, loads).