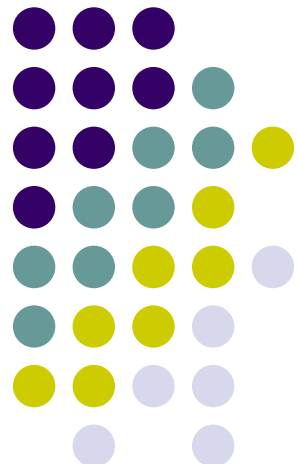
FLP is back!

or

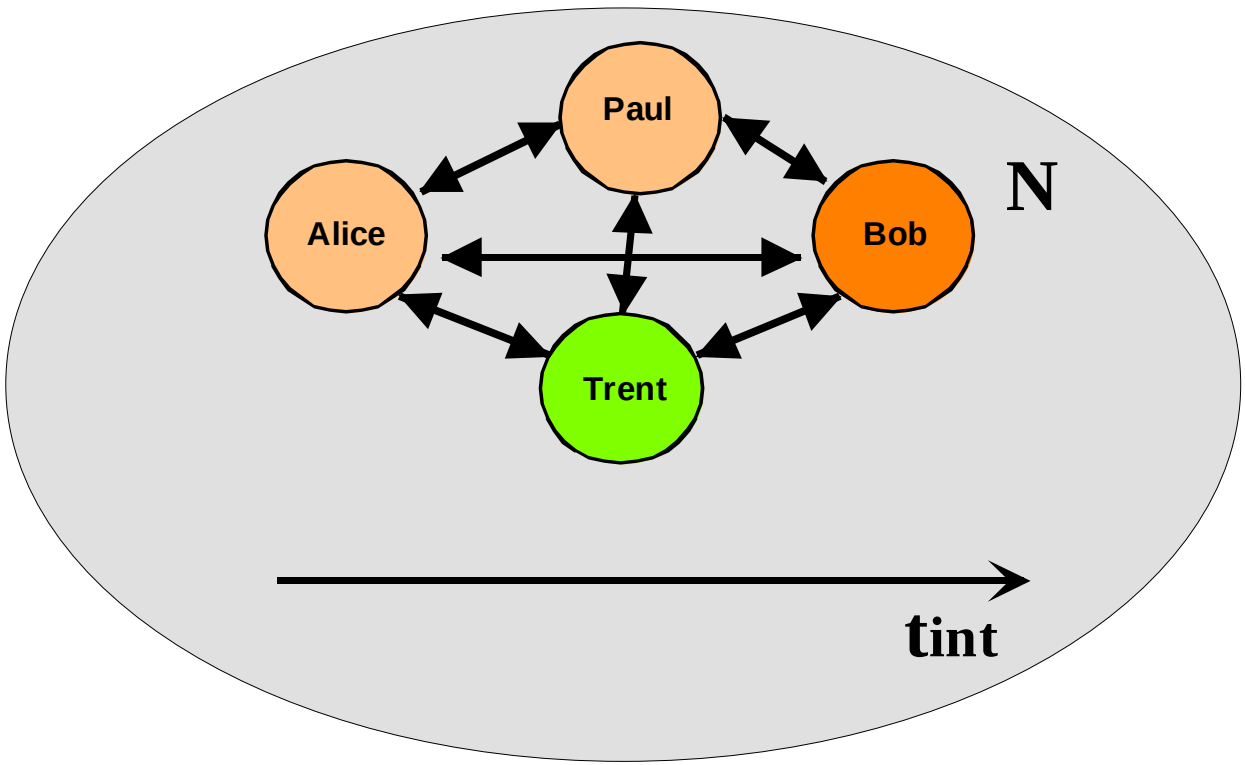
A forgotten dimension of time in distributed systems problems

Paulo Esteves Veríssimo

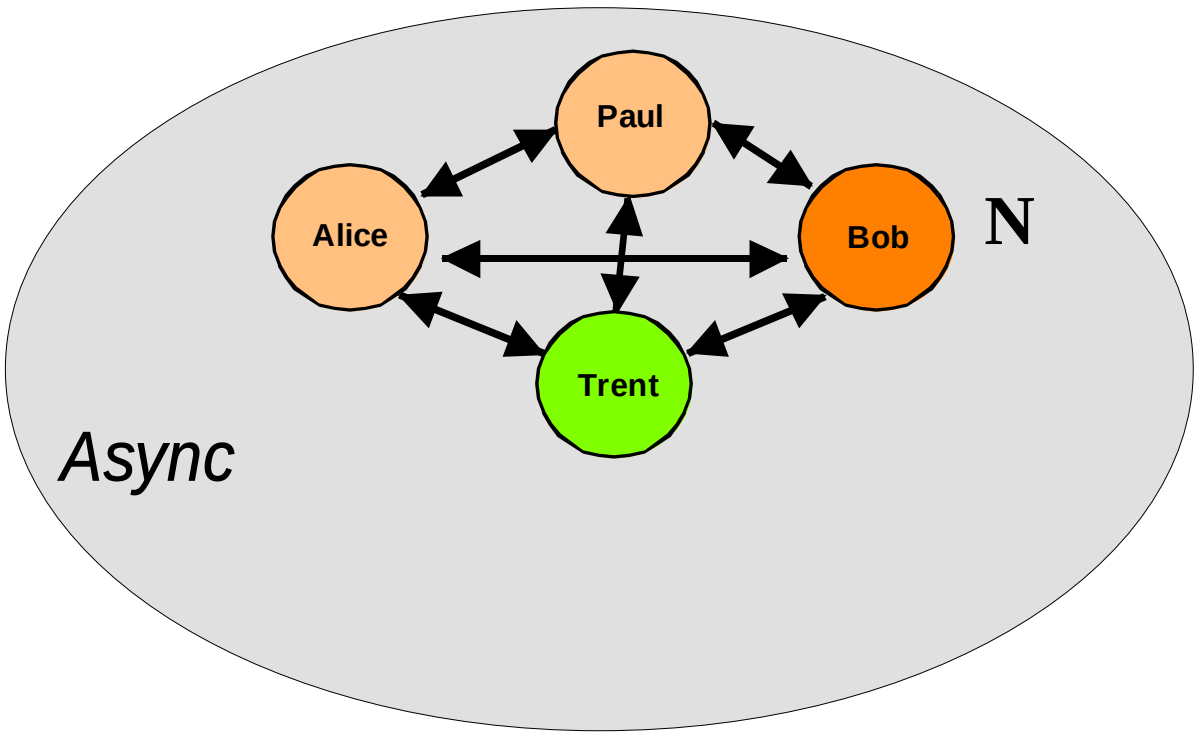
Navigators Group,
LaSIGe, Laboratory for Large-Scale Informatic Systems
Univ. Lisboa
pjv@di.fc.ul.pt
http://www.di.fc.ul.pt/~pjv



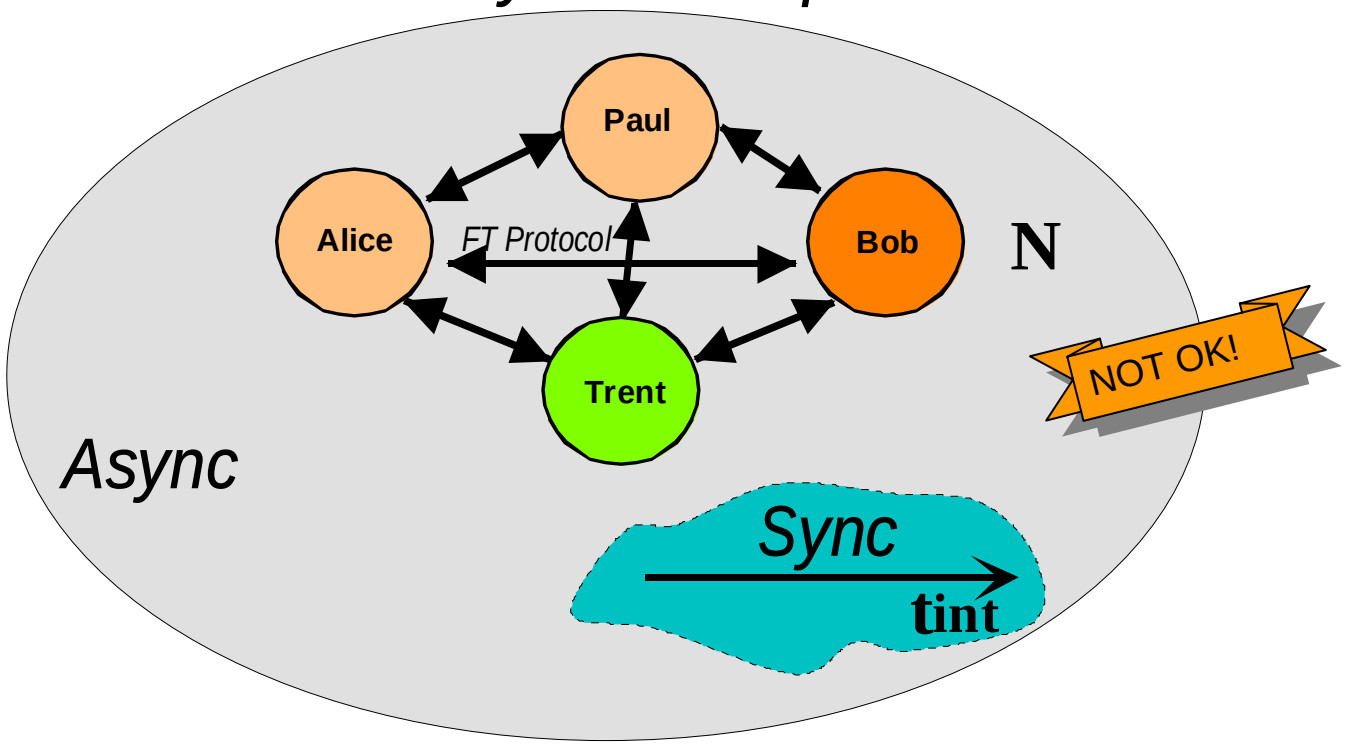
Classical Model



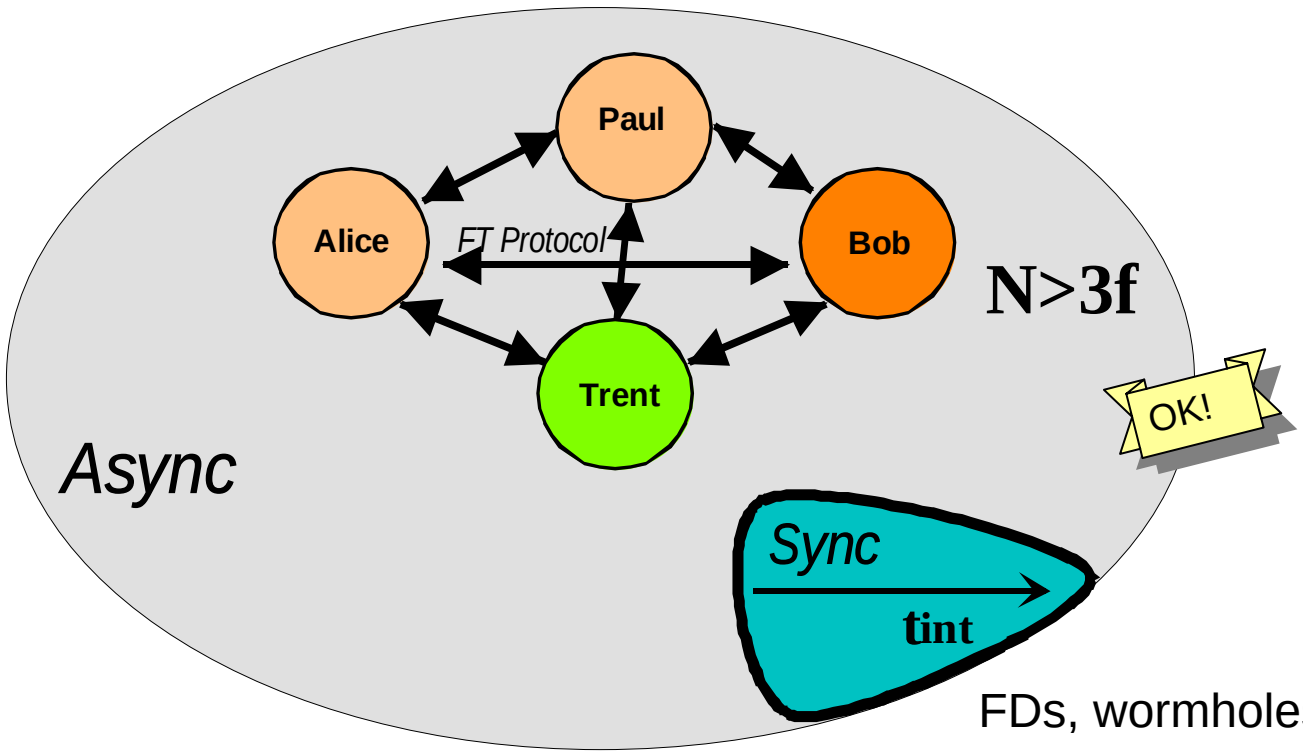
Classical Model - Async System



Classical Model - Async System with hidden sync assumptions

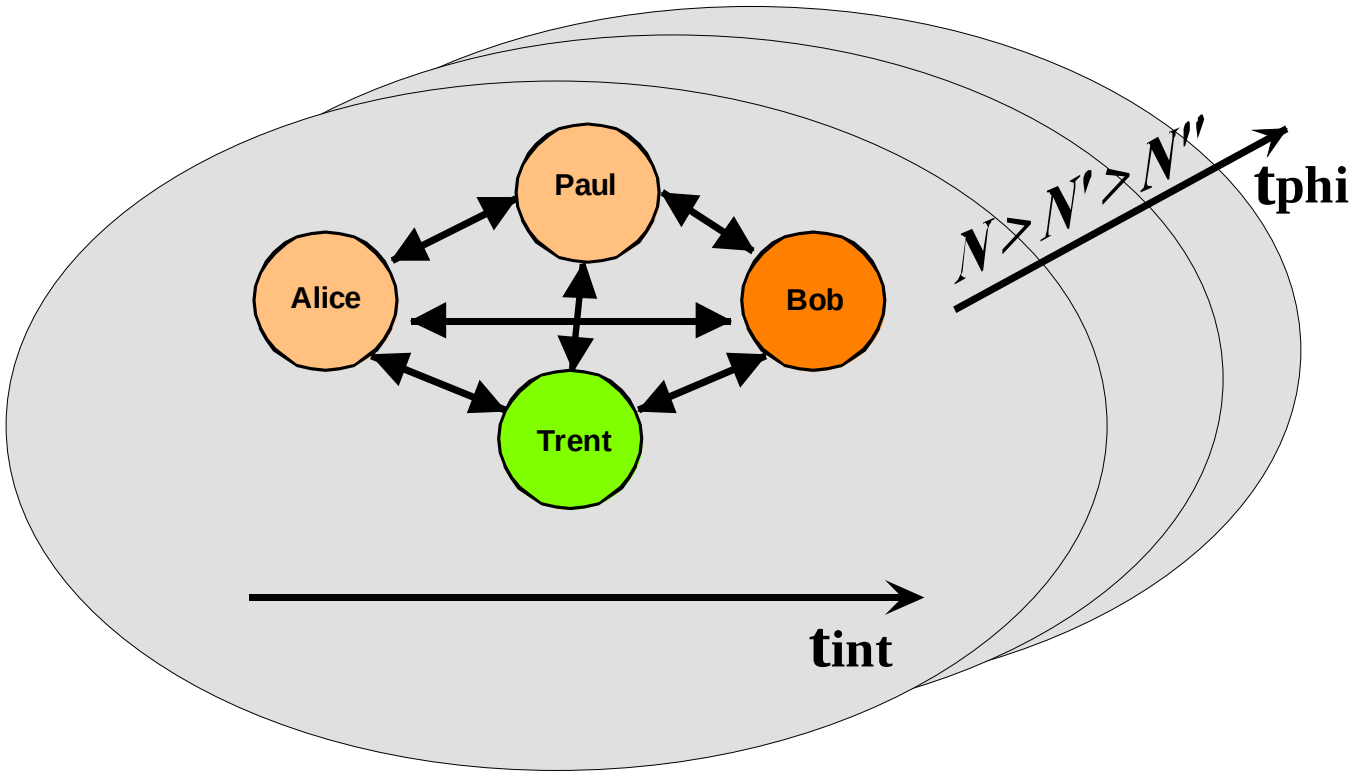


Classical Model - Correct FT Async system

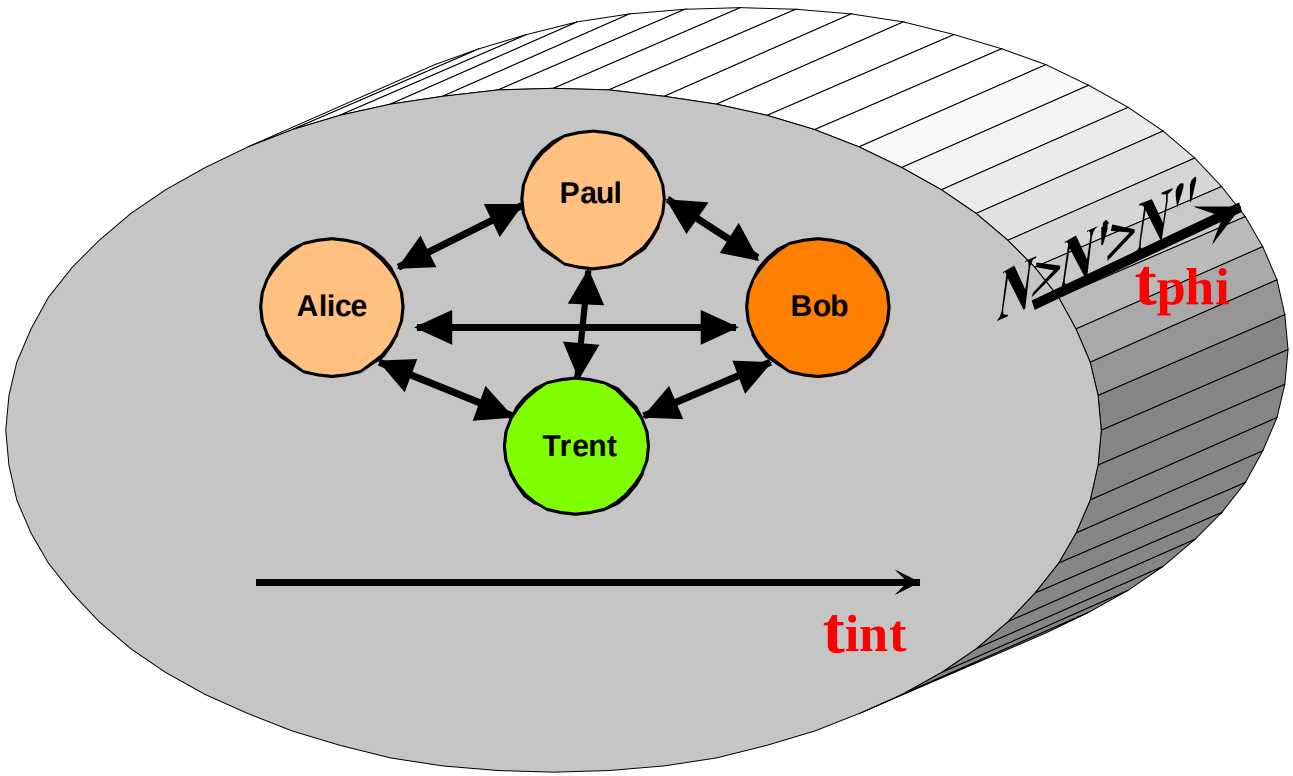


FDs, wormholes, etc

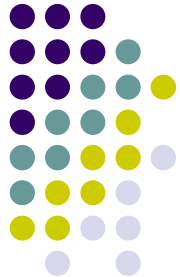
Classical Model vs. Reality



Physical Model

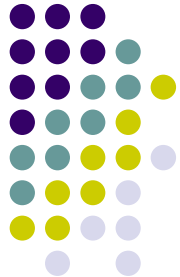


Focusing on Resources



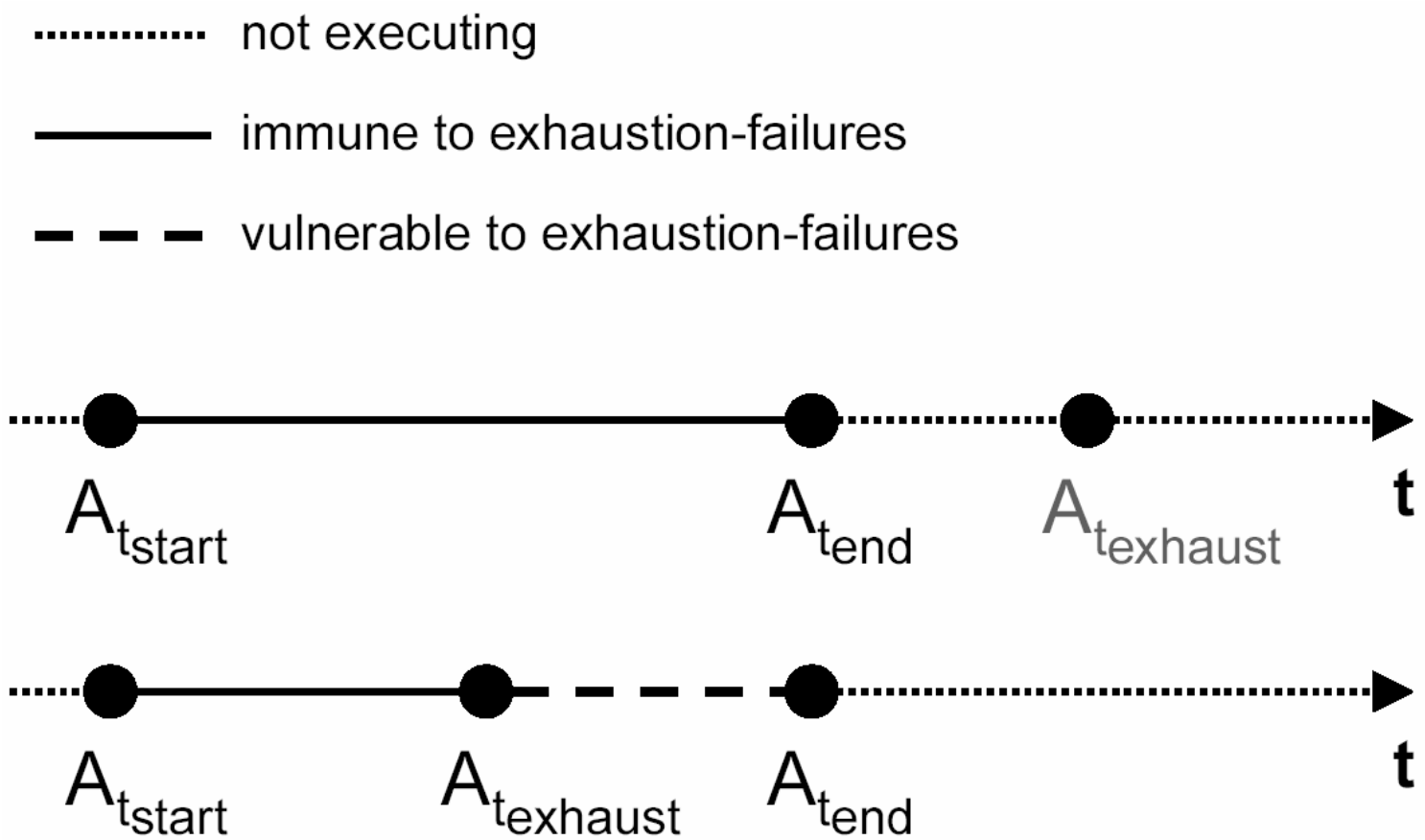
- Fault and timing assumptions are an abstraction of the required resources.
  - e.g., f fault-tolerance means (n-f) correct nodes are required.
- Resource exhaustion: violation of a resource assumption.
  - e.g., f+1 nodes fail.
- Definition: An exhaustion-failure is a failure that results from resource exhaustion.
- Definition: A system is exhaustion-safe if it ensures that exhaustion-failures never happen.

Physical System Model (PSM)



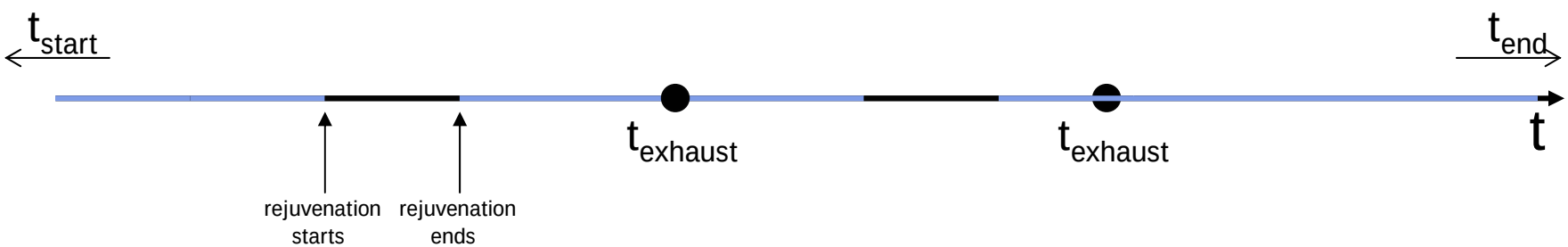
- Allows to formally reason about how exhaustion-safety is affected by different combinations of timing and fault assumptions.
- A system execution is defined by
  - t_{start}: the RT start instant.
  - t_{end}: the RT termination instant.
  - $t_{exhaust}$: the RT instant when exhaustion occurs.
- Definition: A system is exhaustion-safe iff $t_{end} < t_{exhaust}$, for all executions.
  - e.g., a f fault-tolerant distributed system is exhaustion-safe if it terminates before f+1 failures being produced.

To Be or Not to Be Exhaustion-Safe



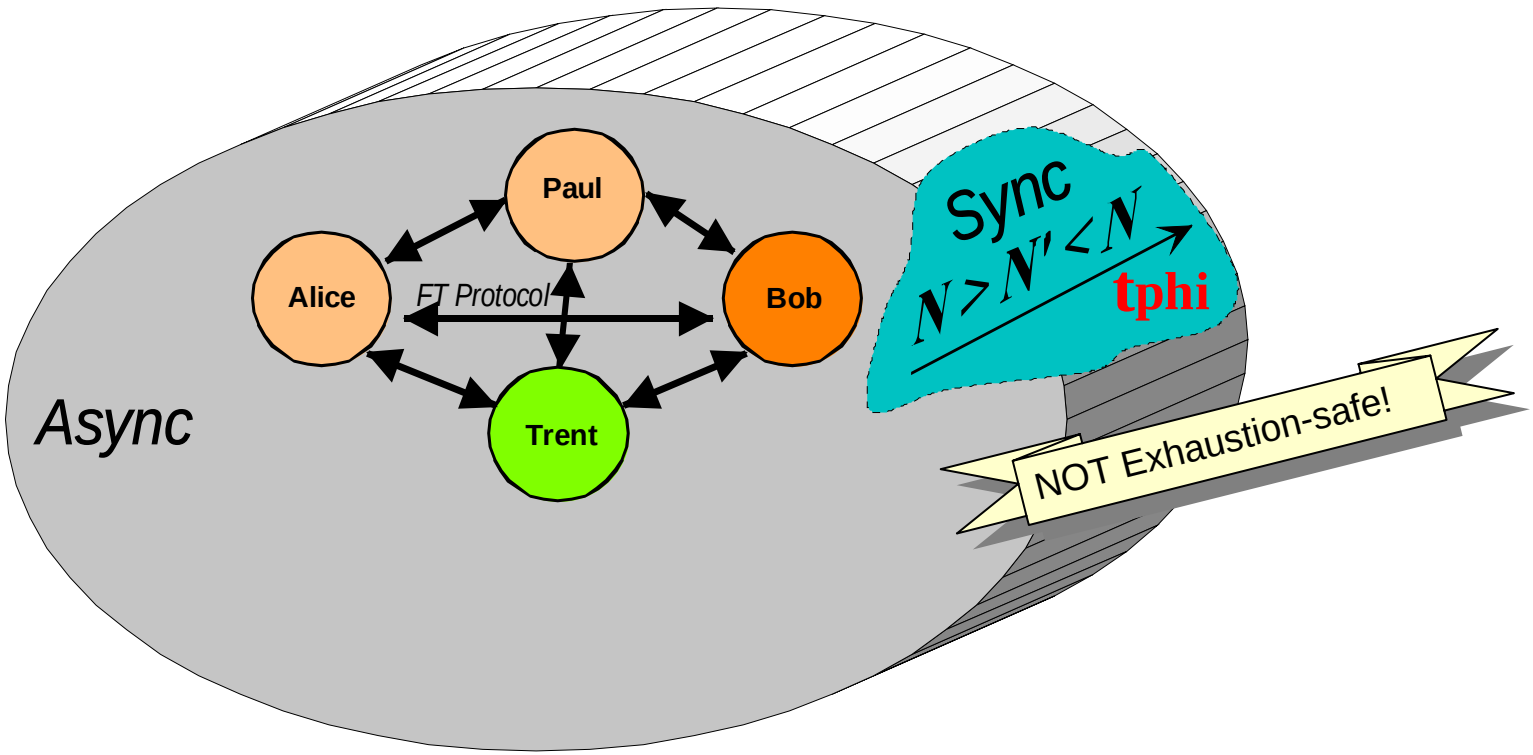
Proactive Recovery

- Goal: to constantly postpone $t_{exhaust}$ through periodic rejuvenation.
  - e.g., periodic rejuvenation of OS code .



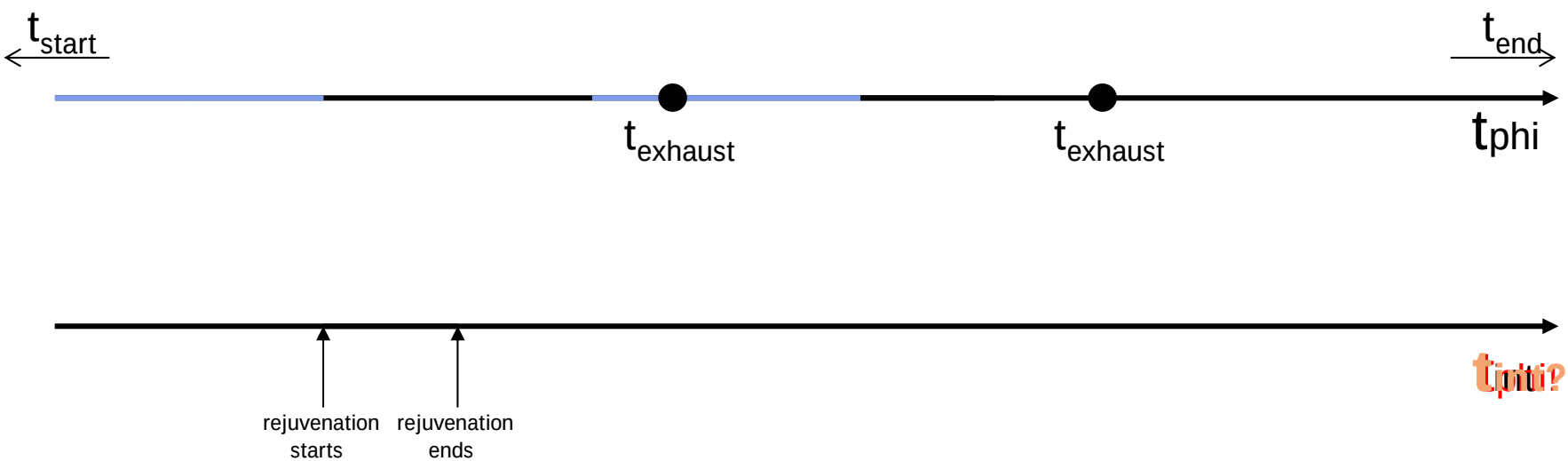
- A system is exhaustion-safe only if rejuvenations are always terminated before exhaustion.

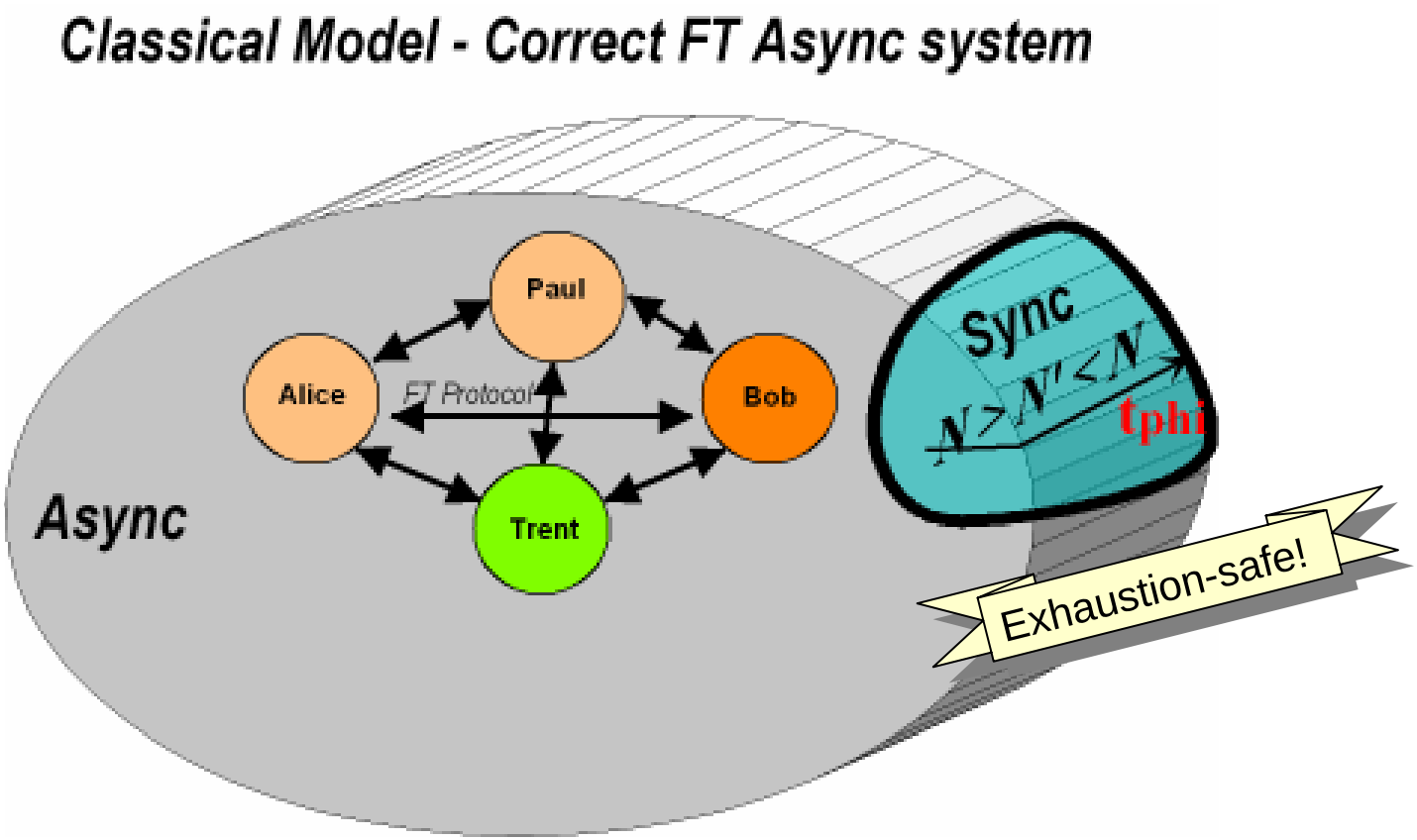
Physical Model - Async system with hidden sync assumptions



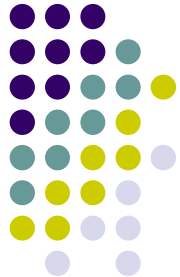
Proactive Recovery

- Goal: to constantly postpone $t_{exhaust}$ through periodic rejuvenation.
  - e.g., periodic rejuvenation of OS code .




Classical Model - Correct FT Async system
Paul
Alice
FT Protocol
Bob
Trent
Async
Sync
N' < N
tphi
Exhaustion-safe!

Conclusions



- Current state-of-the-art does not allow to construct exhaustion-safe distributed systems, specially in face of arbitrary faults:
  - Sync systems are vulnerable:
    - timing failures.
  - Async systems are vulnerable:
    - max number of faults + unbounded execution time.
  - Async systems with async proactive recovery are vulnerable:
    - max number of faults + unbounded rejuvenation period.

Future/Ongoing Work

- Combining proactive recovery and wormholes
  - Proactive recovery is useful to postpone $t_{exhaust}$ as long as it has timeliness guarantees.
  - Proposal: combine async payload system with sync proactive recovery subsystem.
  - See our recent tech report:
    - Proactive Resilience through Architectural Hybridization DI/FCUL TR 05-8, May 2005.
  - http://www.navigators.di.fc.ul.pt/