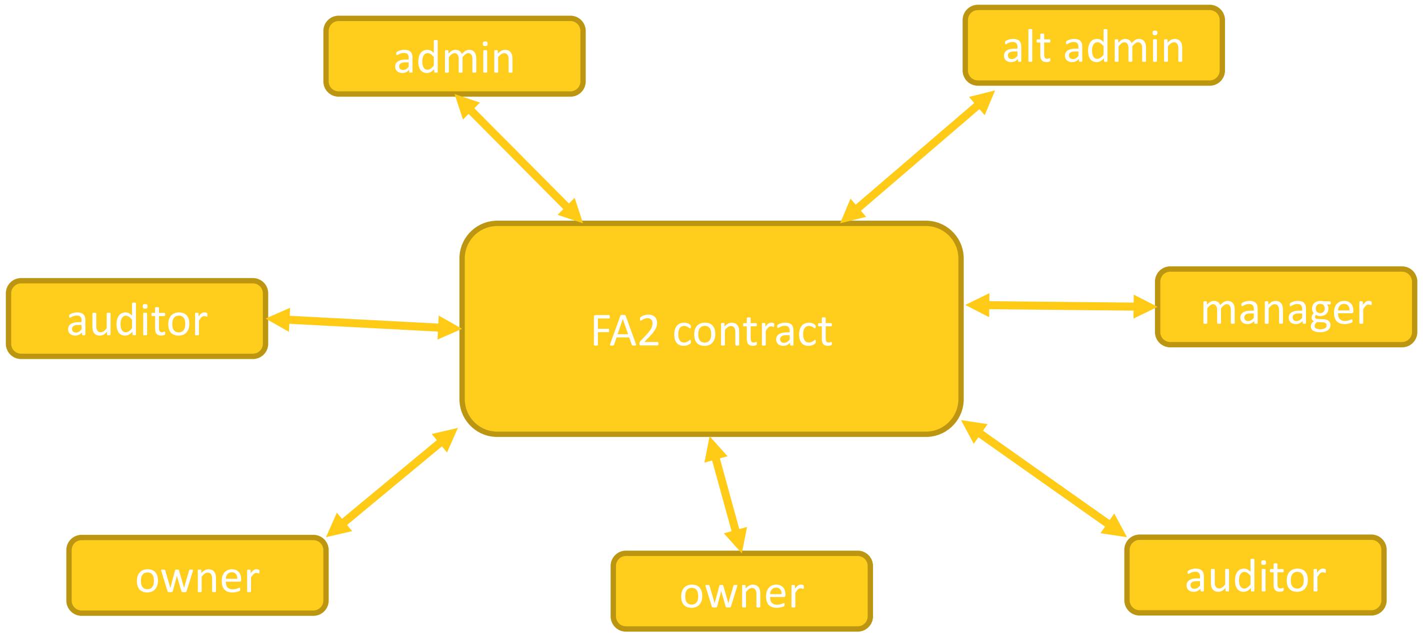
Synchronous Contracts

Georges Gonthier
SmartSpec

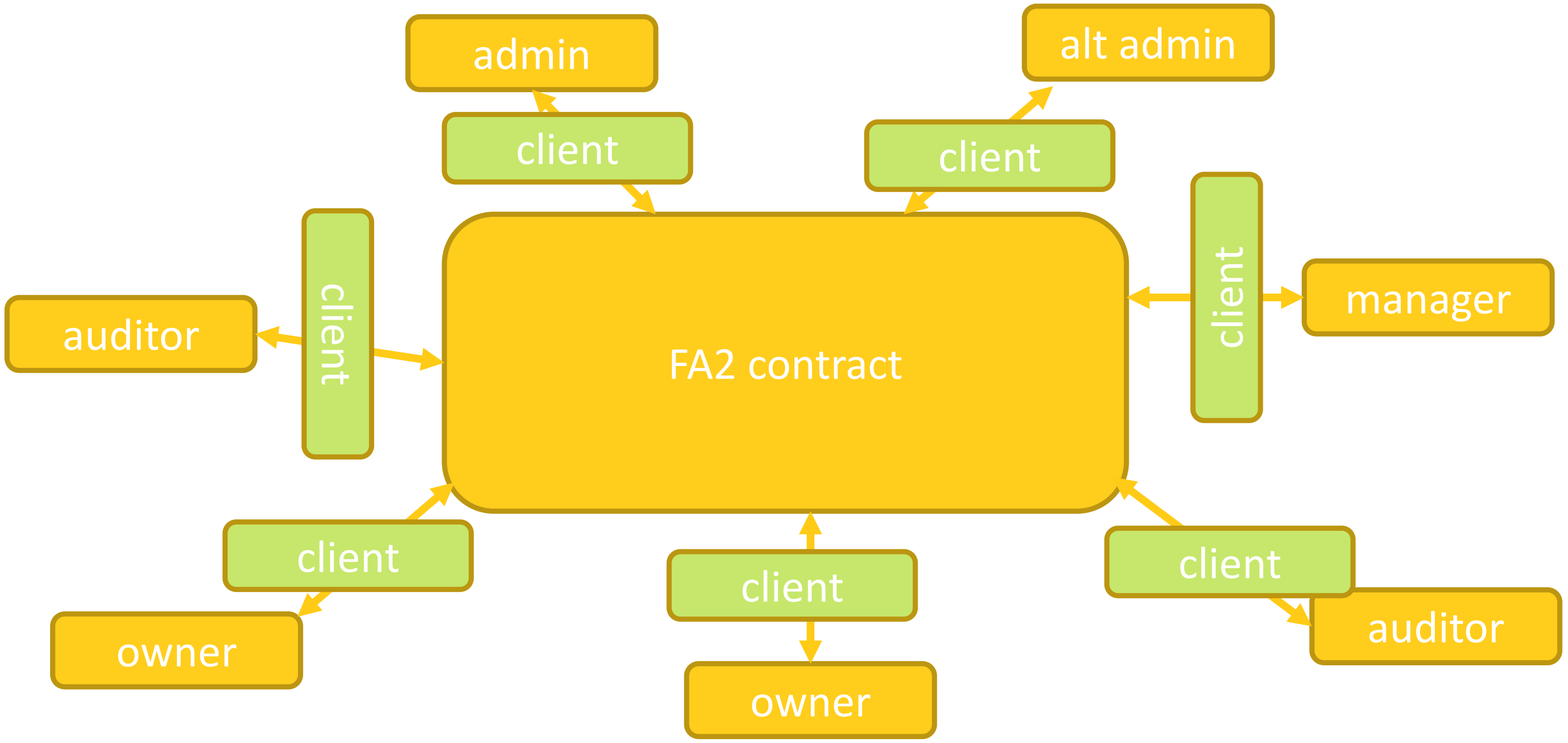
Concurrent services

Concurrency => asynchrony



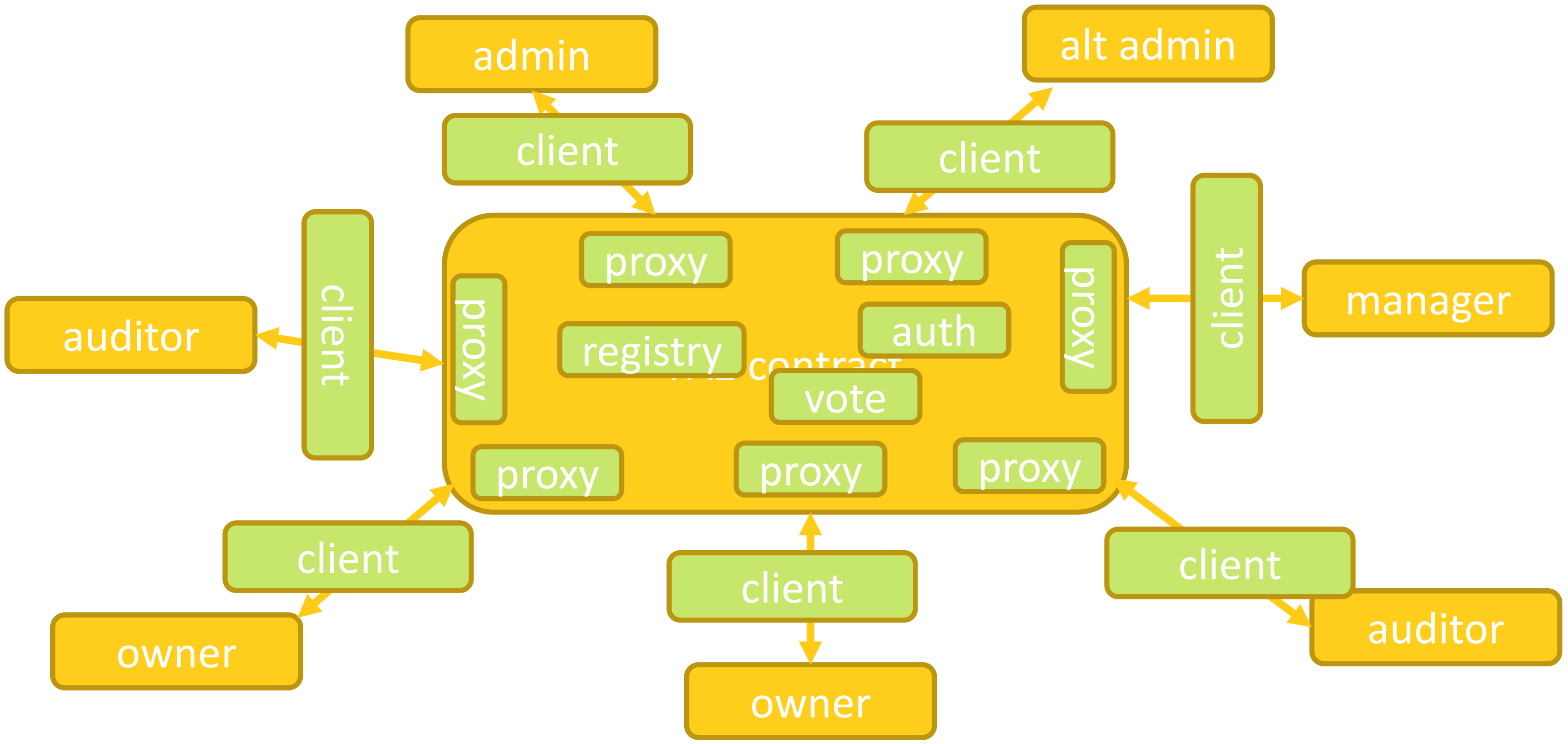
External concurrency

System architecture



Internal concurrency

Software architecture



Asynchonous vs. synchronous

Synchrony

Atomic execution of (global/functional) transactions

- External inputs cannot interleave/interfere with internal message exchange
- Implementation complexity
  - Must schedule execution to avoid deadlock/starvation
- Wrongly conflated with procedural execution
  - Stack scheduling

Asynchrony

Arbitrary interleaving of internal and external messages

- Straightforward implementation and semantics
- Specification complexity
  - Internal architecture exposed
  - Composition / abstraction conflict

Degrees of synchrony

Synchonous procedures

- E.g., Solidity/EVM
- Multiple call conventions
- Asynchrony introduces by callbacks/reentrancy
  - DAO fail

Pure asynchrony

- E.g., Scilla
- Limited function calls
- Descriptive specifications

Scheduled asynchrony

- E.g., Tezos
- Linked transactions
- Fixed scheduling provides some atomicity

In praise of synchrony

Natural semantics: global time

Instant reaction to external inputs / scheduling ticks

- Flag `async` transaction – otherwise instantaneous calls
- Flag `single` transactions that can be called only once per instant
- Test for simultaneous calls with `during t(x, y) { ... }`
  - > Synchronous shared variables can be encoded

Prescriptive specifications

Transparent process refinement

- Atomic transactions can be spread across several processes

Efficient implementation

Static schedule

- Can be verified or compiled
- Can be sliced across distributed processes

Merci !

Suivez-nous sur www.inria.fr