Towards a cross-platform, polyglot implementation of *Aggregate Computing* in ScaFi3

Luca Tassinari

25/03/2025

Motivations

- Aggregate Computing span heterogeneous devices and platforms
- Several implementations of AC exist for different programming languages to:
  - target different platforms and environments;
  - leverage unique strengths of the host programming languages;

However:

- Each of these were developed from scratch, with no code reuse and compatibility in mind;
- No common framework led to fragmentation.

Goal

Investigate the feasibility of building a framework capable of targeting multiple platforms while offering interoperability with other languages.

In particular the work focuses on:

- architectural design of a portable, interoperable layer for Aggregate programming, preserving core abstractions and full code reuse.
- interoperability and distribution strategies enabling seamless data exchange and collective execution across heterogeneous devices and language runtimes;
- evaluation of performance, API idiomaticity, and maintenance effort.

$\Rightarrow$ Scala 3 as the perfect fit to model AC abstractions and model in a strongly typed internal DSL.

Scala 3 compilation targets

- **JVM** (desktop, server, Android) & *Java* interop;
- **JS** via Scala.js:
  - Supported platforms:
    - Web (browser);
    - Node.js;
    - WebAssembly (experimental).
  - *JavaScript* interop via annotations, indirectly supporting *TypeScript*;
  - Mature ecosystem.
- **Native** via Scala Native:
  - Supported platforms:
    - `x86-64` and `aarch64` on Linux, macOS and Windows;
    - experimental 32-bit support;
    - suitable for SoC-based IoT devices but not microcontrollers;
  - *C* interop via annotations;
  - Growing ecosystem maturity, limited toolchain support.

ADD PRO AND CONS

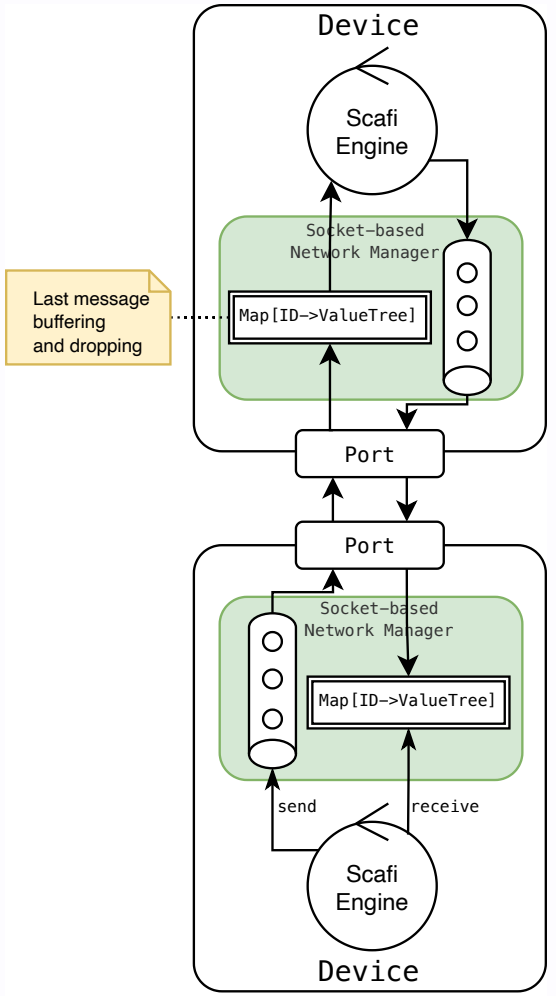
Contribution

The contribution of this thesis span three main axes:

1. **Add a cross-platform *distribution* module;**
2. Add support for a general *cross-platform* and *polyglot* serialization binding;
3. Add a *cross-platform*, *polyglot* library abstraction layer.

1. **Add a cross-platform *distribution* module;**

- Technology: *stream*, *TCP*-based *connection-oriented sockets*;
  - Each device is bound to a specific *endpoint* (IP + port);
  - Point-to-point connections between neighbors;
  - Neighborhood is statically *fixed* at initialization but can be extended in the future with dynamic discovery strategies;
- Support for multiple platforms: *JVM*, *JS* (Node.js), *Native*;
  - *JVM* + *Native* support via Java Standard *sockets* library;
  - *JS* support via *Node.js net* module using Scala.js type facades;
  - **Implications**:
    - shared code cannot perform blocking operations;
    - all the API is designed to be asynchronous and non-blocking using Futures;
    - the primary goal: write as much shared code as possible, minimizing platform-specific implementations.



Simplified class diagram of the socket-based distribution module: