ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea Triennale in Ingegneria e Scienze Informatiche

Decentralized Messaging System with Aggregate Computing

Tesi di laurea in:
Programmazione ad Oggetti

Relatore
Prof. Pianini Danilo

Candidato
Luca Marchi

Correlatore
Dott.ssa Cortecchia Angela

II Sessione di Laurea
Anno Accademico 2024-2025

Abstract

To my family.

Contents

Abstract **iii**

1 Introduction **1**
1.1 Context 1
1.1.1 Aggregate Programming 2
1.1.2 Collektive 5

2 State of the art **7**

3 Contribution **9**
3.1 Fancy formulas here 9

11

Bibliography **11**

List of Figures

1.1 A variety of IoT devices communicating with each other in a real world scenario. 2
1.2 Layered structure of aggregate programming, illustrating the abstraction at each level. 4

List of Listings

listings/HelloWorld.java 9

Chapter 1

Introduction

1.1 Context

In a world where Internet of Things (cro:IoTInternet of Thing (IoT)) devices are becoming increasingly prevalent, the need for efficient and reliable communication systems is paramount Figure 1.1. The interactions between neighboring devices play a crucial role in enabling seamless data exchange and coordination, so there is the need to design networks with infrastructures that support scalability, adaptability and reusability [BPV15]. In the past, it was reasonable to use a programming model that focused on the individual computing device, and its relationship with one or more users. However, as systems have grown in scale and complexity with the number of computing devices rising, this method has become inadequate. Traditional network architectures, rely heavily on centralized infrastructures, making them unsuitable for scenarios such as disaster recovery or interactions with neighboring devices. The computational model of *aggregate computing* provides a promising approach to address these challenges by enabling decentralized and self-organizing systems [VBD+19]. Building on this concept, this thesis explores how aggregate programming can support a proximity and decentralized messaging system in a network.

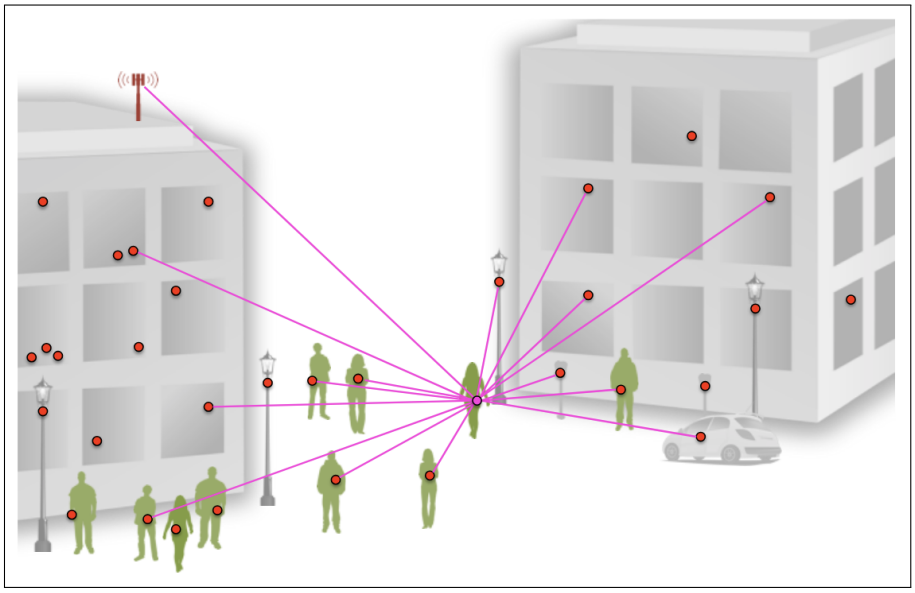

Figure 1.1: A variety of IoT devices communicating with each other in a real world scenario.

1.1.1 Aggregate Programming

Aggregate programming is a distributed systems paradigm that simplifies programming large networks of devices by focusing on the global, system-level behavior rather than the individual behavior of each device. It shifts the abstraction to view the network as an aggregate, an agglomeration of nodes that collectively exhibit certain behaviors. This model is particularly well-suited for scenarios where devices need to coordinate and collaborate based on local interactions, such as in IoT applications. As discussed in distributed field programming [VBD+19], there are principles to follow when designing large-scale systems:

- The mechanisms ensuring robust coordination should operate transparently in the background, so that programmers don't need to manage them directly.
- The composition of modules and subsystems should be simple and predictable, allowing developers to clearly understand the consequences of combining components.
- Large-scale distributed systems often consist of multiple heterogeneous subsystems, each of which may require different coordination strategies depend-

ing on the region or context in time.

Aggregate programming aims to overcome these challenges through three fundamental principles:

1. The computational target is conceived as a region of the environment, with the underlying device-level details abstracted away, potentially even modeled as a continuous spatial domain.

2. The program logic is expressed as the manipulation of data structures that extend spatially and temporally across that region.

3. These computations are executed locally by individual devices within the region, which coordinate through resilient mechanisms and proximity-based interactions.

The foundation of this paradigm lies on *field calculus* [BPV15], a core set of constructs modeling device behavior and interaction. These essential features are captured in a tiny universal language suitable for mathematical analysis. The concept of **field** plays a central role in this context and originates from physics, where it is defined as a function mapping every point in a space–time domain to a scalar value. According to [VBD+19], the *field value* ϕ is a function

$$\phi : D \to L$$

that maps each device δ in the domain D to a local value ℓ in the range L. This value can change in time with a *field evolution* that maps each point in time to a field value. Figure 1.2 shows how aggregate programming abstracts the complexity of the underlying distributed network and its coordination challenges through a sequence of hierarchical abstraction layers.

The messaging system developed in this thesis was implemented using *Collektive* [Cor], an open-source framework for aggregate computing written in Kotlin. This framework provides the developer-facing APIs (shown in Figure 1.2) used to define and execute aggregate applications.

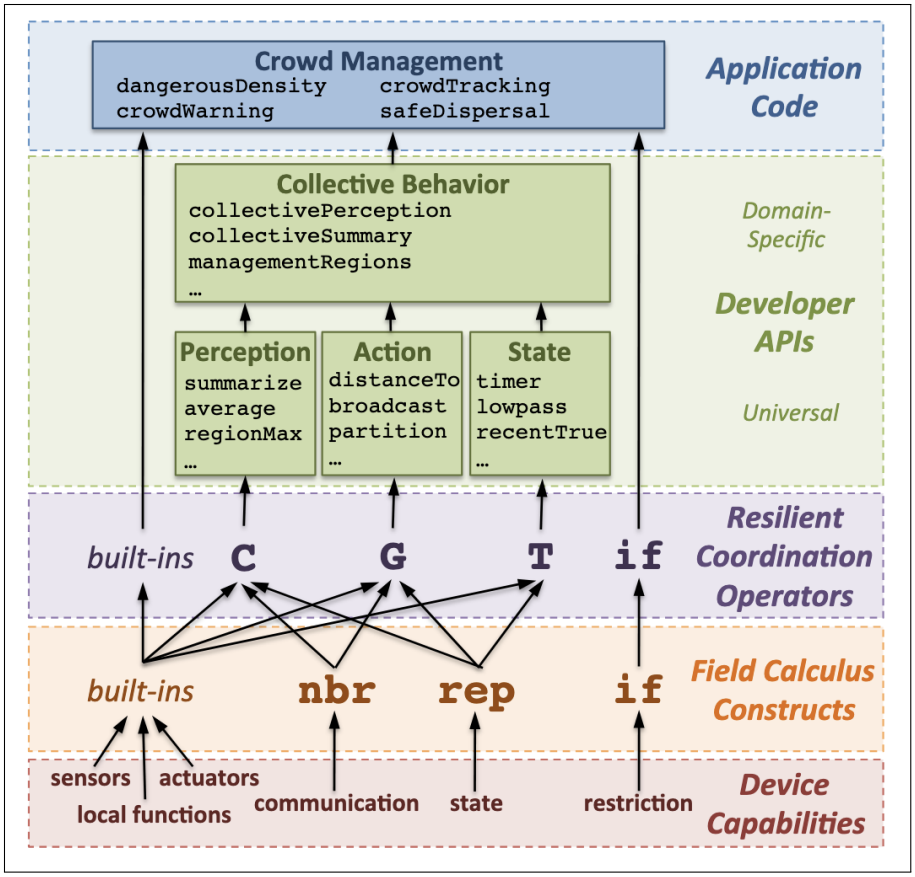

Figure 1.2: Layered structure of aggregate programming, illustrating the abstraction at each level.

1.1.2 Collektive

Collektive

Chapter 2

State of the art

Chapter 3

Contribution

You may also put some code snippet (which is NOT float by default), eg: chapter 3.

3.1 Fancy formulas here

```
public class HelloWorld {
  public static void main(String[] args) {
    // Prints "Hello, World" to the terminal window.
    System.out.println("Hello, World");
  }
}
```

Bibliography

[BPV15] Jacob Beal, Danilo Pianini, and Mirko Viroli. Aggregate programming for the internet of things. *Computer*, 48(9):22–30, 2015.

[Cor] Angela Cortecchia. *A Kotlin multi-platform implementation of aggregate computing based on XC*. PhD thesis.

[VBD+19] Mirko Viroli, Jacob Beal, Francesco Damiani, Stefano Montagna, Danilo Pianini, Luca Ricci, and Franco Zambonelli. *Aggregate programming: from foundations to applications*. Springer, 2019.