University of Magdeburg

School of Computer Science

Bachelor Thesis

Mixed-reality Simulation of Quadcopter-Swarms

Author:

[Lukas] [Mäurer]

Matrikelnummer: 189073

September 30, 2015

Advisors:

Prof. Sanaz Mostaghim

Department of Intelligent Systems

Christoph Steup

Department of Intelligent Systems

***[Mäurer]*, *[Lukas]*:**
Mixed-reality Simulation of Quadcopter-Swarms
Bachelor Thesis, University of Magdeburg, 2015.

Contents

List of Figures

List of Tables

List of Code Listings

1. Introduction

1.1 Motivation

[project context of this work]

[who needs the results]

[what are the problems to be solved?]

[what are existing solutions, what's different in this approach, what is the improvement]

1.2 Problem Statement

[what are the goals] [how are we going to reach this goals] [what is to be done]

1.3 Outline

[short description of the sections]

2. Theory

2.1 Quadcopter Modelling

[fundamental physics]

[particle simulation]

2.2 Vrep

[connecting visual representation and physical model]

[simulation structure (lua scripts, scene structure]

[lua module structure]

[external interface (signals)]

2.3 Communication V-REP-Quadrocopter

***Goal*:** our mixed reality simulation needs a dependable link of communication between the V-REP simulation environment and the flying quadrocopters. The Quadcopter needs to stream its telemetry data in real-time to the V-REP, and the reverse communication is needed as well. The communication between the V-REP and the quadrocopters passes through several software components, which are depicted on figure Figure 2.1.

Remote API: V-REP provides several means of communication with an external application. One of them is the Remote API, which allows to control a simulation (or the simulator itself) from an external application or a remote hardware (e.g. real robot, remote computer, etc.). The V-REP remote API is composed by approximately one hundred functions that can be called from a C/C++ application, a Python script,

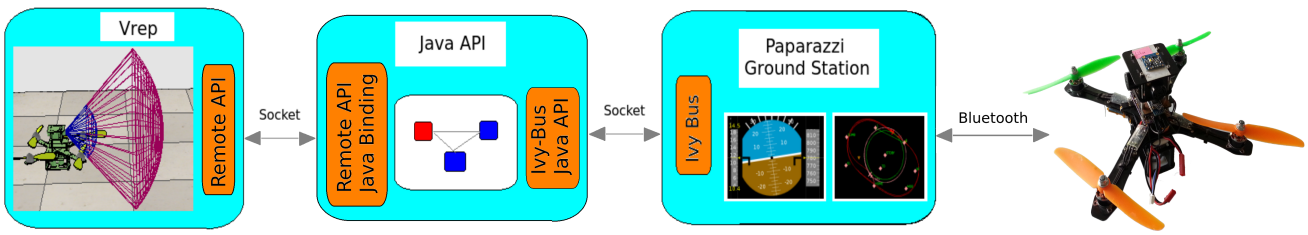

Figure 2.1: communication V-REP - Quadrocopter

a Java application, a Matlab/Octave program, an Urbi script, or a Lua script. The remote API functions are interacting with V-REP via socket communication in a way that reduces lag and network load to a great extent.

Java API is the external program, that communicates with V-REP through the Remote API. We have chosen to implement our external program, communicating with the V-REP, in the Java programming language regarding the following advantages: Java's platform independence allows to run the external program even on different machine with different operating system than the one used for running the V-REP environment. Java is object-orientated which favours the use of design patterns and highly abstraction layers, which allows us to write an API that is modular, reusable and can later be easily extended to support other mixed-reality scenarious. The implementation and architecture of the Java API is duscussed in details in Chapter 3. The purpose of the Java application is to serve as a communicating bridge between the Paparazzi Ground Station and the V-REP. It detects all quadrocopters in the V-REP simulation, builds their virtual representations and feeds the models with real-time data.

Ivy Bus is a simple protocol and a set of open-source (LGPL) libraries and programs that allows applications to broadcast information through text messages, with a subscription mechanism based on regular expressions. Ivy libraries are available in C, C++, Java, Python and Perl, on Windows and Unix boxes and on Macs. The Paparazzi Ground Station uses the Ivy Bus as a means of communication between the different software components. Figure Figure 2.2 depicts the communication structure in the Paparazzi Ground Station, in which the different agents communicate with each other by sending messages on the Ivy-Bus.

The UAV (in blue) is streaming its telemetry data to the ground control station, which is received by the link. The **link** agent manages the ground-based radio modem and distributes the received messages to the other agents across the Ivy-Bus.

The Ivy Bus is an example of a publisher-subscriber protocol, in which senders of messages, called publishers, does not explicitly specify the address of the receiver, but just send the message on one, shared by all nodes, bus. The recipients, called subscribers, which are interested in the message will accept it and the others will ignore it. The publisher-subscriber is a many to many communication model in which publishers are

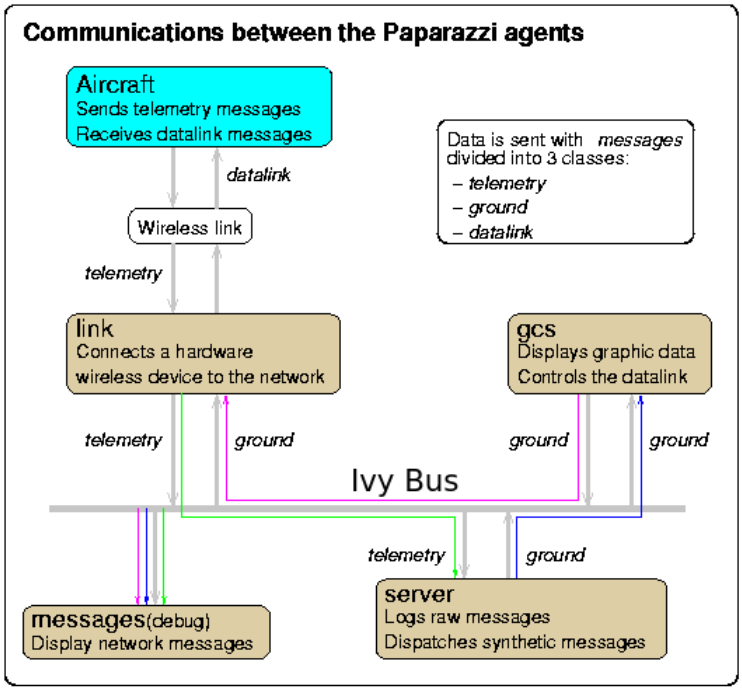

Figure 2.2: Ivy-Bus in Paparazzi Ground Station

loosely coupled to subscribers - there is no space, flow and time coupling. This means that the publishers does not have to know the addresses of the subscribers and event does not need to know of their existence. Each can operate normally without the other and can continue its thread of execution regardless if the subscriber has received the message or not. It also provides scalability, which means that we can “attach” our Java API to the Ivy-Bus and start publishing and listening for messages without changing any line of source code in the Paparazzi Ground Station software.

In the publisher-subscriber model, subscribers typically receive only a subset of the total messages published. The process of selecting messages for reception and processing is called filtering. There are two common forms of filtering: topic-based and content-based. In a topic-based system, messages are published to ”topics” or named logical channels. Subscribers in a topic-based system will receive all messages published to the topics to which they subscribe, and all subscribers to a topic will receive the same messages. The publisher is responsible for defining the classes of messages to which subscribers can subscribe. In a content-based system, messages are only delivered to a subscriber if the attributes or content of those messages match constraints defined by the subscriber. The subscriber is responsible for classifying the messages. Both filtering techniques are depicted in figures Figure 2.4 and Figure 2.3.

The Ivy Bus is a content-based publisher-subscriber and uses regular expressions for the message filtering.

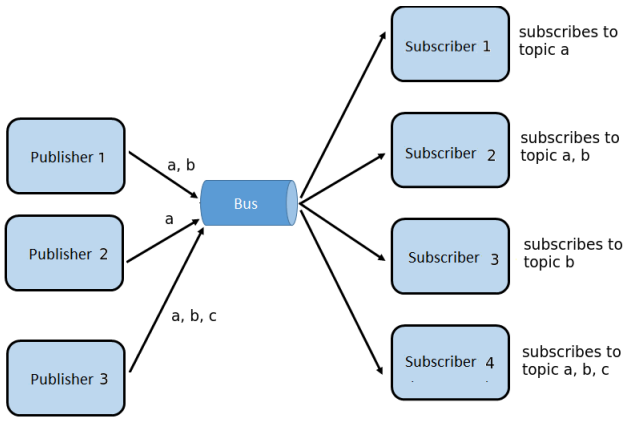

Figure 2.3: Topic-based publisher-subscriber

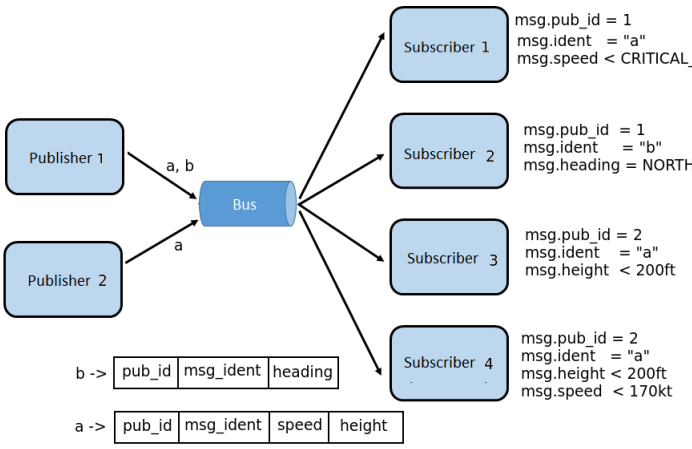

Figure 2.4: Content-based publisher-subscriber

3. Implementation

3.1 Simulation Environment

[finken parameter estimation]

[controller tuning]

[simulation parameters]

[script structure]

3.2 Communication Link

[link in Vrep (signals]

3.3 *[fancy name]*

3.4 Quadcopter

4. Evaluation

[how realisitc is the simulation?] [which properties can be modelled well, which can't?]

4.1 Speed

[communication delay] [vrep simulation speed]

4.2 Accuracy

[error]

4.3 |

Stability

4.4 usability

5. Conclusion

5.1

[do the results show that it works?]

5.2 Future Work

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Magdeburg, den 29. November, 2013