The Bottleneck Problem
Distributed Systems - Project 2019

Edoardo Lenzi, Talissa Dreossi
DMIF, University of Udine, Italy

Version 0.1, May 22, 2019

Abstract

The aim of this project is the analysis, the implementation and testing of a distributed solution for the **bottleneck problem** (also know as *The Monkeys Problem*, in operative systems theory).

Basically the problem consists in a two-way road with a **bridge** (bottleneck) in the middle, the bridge has a certain **maximum capacity** and can be crossed in only **one direction at a time**.

The bridge is **very risky** cause it is located in a remote place where there is nothing that can prevent **car crashes/congested traffic** but cars.

Note that **cars** here are **autonomous systems** without any human driver inside and can send messages with others adjacent cars with some wireless technologies (ie. bluetooth, wifi, ...) in order to solve the situation.

The projet requires the implementation of a **simulator/business logic** for the environment setup and a **web server** that will expose the simulation state with some API for any further **UI application**.

Chapter 1

Introduction

In this chapter we are going to describe accurately the problem and provide a possible solution (from the point of view of a Distributed Systems designer).

1.1 The Problem

The bridge has:

- **maximum capacity** $c \geq 1$

- **length** $l \geq 1$

Cars can send messages to adjacent cars (to the car in front and to the rear one); cars can also have the ability to speak with more than two other cars depending on the **power of their transmission medium** $p \geq 1$.

So basically a car can only send messages to the p cars in front and to the p cars behind. We assume that the bridge length doesn't constitute an impediment for the communications (so even if the bridge was very long, however, the cars can send messages as if they were close).

Moreover a car has a certain **speed** $s \geq 1\frac{block}{s}$ that determines the bridge crossing time and the time to reach the first obstacle (another enqueued car or the bridge).

For simplicity we assume that every car has the same dimensions expressed in an arbitrary length scale called **block** and **every measure is an integer** (s, l, c, p, etc.).

$\boxed{c = l = 2}$

$\boxed{\begin{array}{l} p = 1 \\ s = 1\frac{block}{s} \end{array}}$
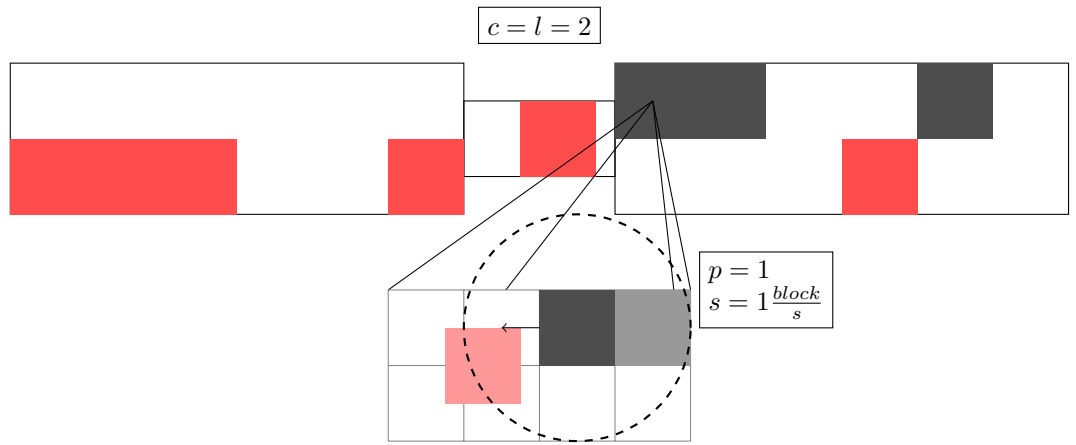
Figure 1.1: Example of a possible situation

Chapter 2

Analysis

In this chapter, we describe in detail functional and non-functional requirements of a solution for the problem.

2.1 Functional requirements

Which functions must be offered to users / other programs? Which are the input data and the output data? Which is the expected effect?

2.2 Non functional requirements

Everything about mode and transparencies: availability, mobility, security, fault tolerance, etc.

Are there execution time bounds? Minimum data rates?

If requested, specific platforms/languages/middlewares requirements for the implementation can be decided here. (E.g.: if the project is on a SOA, we may request that functions are offered via SOAP or RESTful services).

Chapter 3

Project

This chapter is devoted to the description of the general architectures, and specific algorithms.

3.1 Logical architecture

Describe the components of your systems: modules/objects/components/services. For each component, describe the functionalities it implements, and by who is used.

3.2 Protocols and algorithms

Communication between components. UML sequence diagrams go here.

Also, put here a detailed description of distributed algorithms used to solve specific problems of the project.

3.3 Physical architecture and deployment

Which nodes and platforms involved, and where each component is deployed.

3.4 Development plan

Since it is difficult to predict just how hard implementing a new system will be, you should formulate as a set of "tiers," where the basic tier is something youre sure you can complete, and the additional tiers add more features, at both the application and the system level.

Chapter 4

Implementation

Details about the implementation: every choice about platforms, languages, software/hardware, middlewares, which has not been decided in the requirements.

Important choices about implementation should be described here; e.g., peculiar data structures.

Chapter 5

Conclusions

What has been done with respect to what has been promised in Chapters **??** and **??**, and what is left out.

Appendix A

Appendix

In the Appendix you can put code snippets, snapshots, installation instructions, etc.

Evaluation