

Large-Scale Distributed Systems: Exercise Session 2

Joeri Hermans
joeri.hermans@doct.ulg.ac.be

October 24, 2017

1 Introduction

Broadcast communication is used to disseminate information among a set of processes and differ according to the reliability of the dissemination. In a traditional *server-client* architecture, interactions are often established between two processes. In this architecture, a server exposes some kind of interface to the client, which in turn exploits this interface to exchange some state with all other participating clients (e.g., as in a multiplayer game). These interactions are supported by a *direct link* between the server and all its clients. Meaning, there is a dedicated *point-to-point* link for every client with the centralized server.

Of course, this approach has several advantages and disadvantages. The main, albeit practical advantage of this paradigm is that only a single machine (or a set of machines that appear as a single service) is responsible for maintaining some state. This enables system engineers to fine-tune their service without relying on other clients which may have non-optimal connections, or have even malicious intents. This is actually what we observe nowadays. We are moving from a once idealized notion of *peer-to-peer* content delivery systems to *cloud* based architectures (NetFlix, YouTube, ...). Of course, a very high cost is associated with building such a system. However, it provides consistent performance because it does not have to rely on the *quality* of participating nodes in the network. Nevertheless, it brings the traditional issues of introducing a single point of failure for the system. So a paradigm shift of *expecting failure* in P2P computing towards *preventing failure* can be reconsidered. Of course, these monolithic services still consist of several thousands of individual, but cooperating machine. So in a sense you still have to *expect failure* within this architecture.

On the other hand, in *multiparticipant systems* broadcasting is slightly more complex since all participating nodes have to agree on the message they collectively received. Additionally, there is the issue how a message is efficiently distributed to all nodes in the network. Furthermore, one might as well be interested in the ordering of broadcasted messages. To accomplish this, it is useful to rely on *broadcasting abstractions*.

2 Exercises

Exercise 1

What is Best-Effort Broadcasting? Does it ensure the delivery of a message? If so, what is the mechanism ensuring the delivery of the message? Describe an architecture (including requirements) which implements Best-Effort Broadcasting.

Exercise 2

Does duplication of messages occur in Best-Effort Broadcasting? Why / Why not?

Exercise 3

Describe Reliable Regular Broadcasting. How does it differ from Best-Effort Broadcasting? What is the additional liveness property that is included in the abstraction to ensure reliability? How does this additional property ensure reliable delivery?

Exercise 4

Lazy Reliable Broadcasting and Eager Reliable Broadcasting are instances of Reliable Regular Broadcasting. Describe both approaches and how they differ from each other.

Exercise 5

In order to ensure reliable broadcasting, Lazy RB and Eager RB employ a filtering mechanism to ensure that a particular message is not delivered twice to the upper-layer software. How would you design such a filtering mechanism?

Exercise 6

What is implied with Uniform Agreement? Give an high-level overview of the strategy required to implement such a requirement.

Exercise 7

Is the total ordering of messages guaranteed in reliable broadcasting? Why / Why not? What type of mechanisms are required in order to construct an ordering over the messages assuming that the number of involved processes do not change. Why?

Exercise 8

How does Ethernet (IEEE 802.3) relate to broadcasting? If Ethernet supports broadcasting, does the technique ensure reliable broadcasting? In the other case, how would you implement (reliable) broadcasting with Ethernet using the techniques you saw in class? Note: One could apply the same mechanisms to WiFi due the the presence of a shared medium. However, Ethernet is a little bit more tricky since not all machines are connected to the same physical cable.

Exercise 9

Describe gossiping, and how it relates to epidemiology.

3 Assignment

In about two to three weeks we introduce the projects which you will have to implement. One of which, requires you to dome some analysis using Apache Spark and construct a reproducible workflow using Jupyter Notebooks. I recommend that you install the Anaconda Python distribution (<https://www.anaconda.com/download/>), and get familiar with the Jupyter environment. Additional, and more detailed instructions will follow.