# Algoritmos e Sistemas Distribuídos- Project 1 (Default Variant)

João Carlos Antunes Leitão

NOVA Laboratory for Computer Science and Informatics (NOVA LINCS)

and

Departamento de Informática

Faculdade de Ciências e Tecnologia

Universidade NOVA de Lisboa

V 1.0 $\alpha$

$30^{th}$ September 2018

## 1 Overview

This document discusses the first ASD project for 2018/19. There are three options: A, B, C. These options all evaluate for a final project grade of 20. They differ both on the concrete objectives and on the number of students in the group. Regarding this last aspect, option A aims at groups composed of 3 students, option B targets groups of 2 students; and finally, Option C aims at allowing students to pursue an individual project that can has additional freedom.

For Option A and B students are free to use any programming language of technologies they feel more comfortable with. The concrete suggestion is for students to use the Scala/Akka ecosystem. Using Akka, you probably want to have each process in the system to be modeled as a set of Actors, where each actor represents one protocol used by that process. Notice that, you cannot make use of any mechanism provided by the runtime to get knowledge about the system membership. Java is also another option that students, although this might bring some additional complexity to your implementation. Students doing options A or B should discuss with the course professor before starting to use different programming languages.

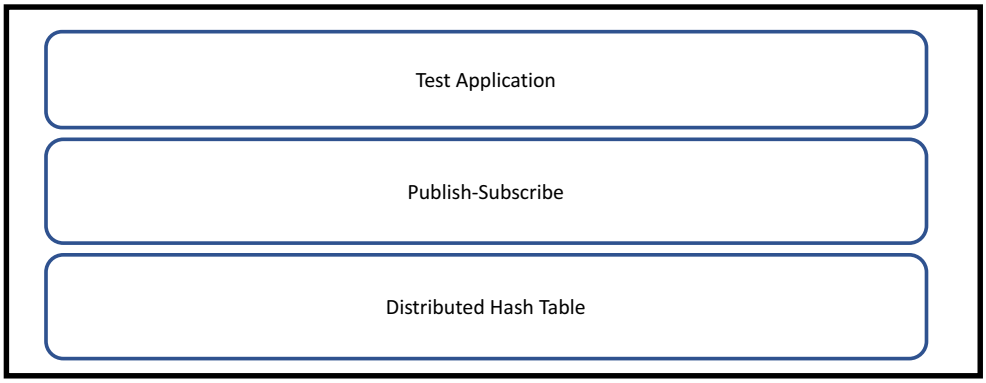
Option C is fundamentally different. This option should be taken by students that aim at leveraging this project to either explore a particular type of distributed protocol (outside the boundaries of the Byzantine fault model). This is suggested only to students that already have a concrete idea of what they might be going to pursue in the context of the master thesis, so that the project can be aligned with that. Additionally, it is highly recommended that students electing option C conduct the project in the context of the Yggdrasil framework presented in the second laboratory class, which implies being very knowledgeable on C programming.

**Evidently, each student should only do one of the available options for the project**

Below each of the options is discussed.

## Part I

# Option A: Publish-Subscribe using Structured Approaches




Representation of a Node in the System

In this option the students are asked to implement a publish-subscribe protocol on top of a distributed hash tables (DHT, also known as structured overlay network).
As we will discuss in the lectures, a DHT organizes nodes according to a logical identifier in a decentralized fashion. The interface exposed by the DHT protocol to other (local) protocols is quite simple, and composed of a single operation denoted, in the context of this project, as `Route( identifier, message )` that delegates on the protocol the responsibility of delivering a `message` to the node, currently in the system, whose identifier is closest to `identifier` and strictly preceding it (considering a circular identifier space.
The publish subscribe system, offers to other protocols the following interface:

`subscribe ( TOPIC )`: That notifies the system that the protocol above is interested in received (through a notification) all messages published that are tagged with `TOPIC`.

`unsubscribe ( TOPIC )`: That notifies the system that the protocol above is no longer interested in receiving messages tagged with `TOPIC`âĂę

`publish ( TOPIC, message )`: That publishes, to the system, a `message` tagged with `TOPIC`.

For simplicity, we consider that messages are tagged with exactly one topic.

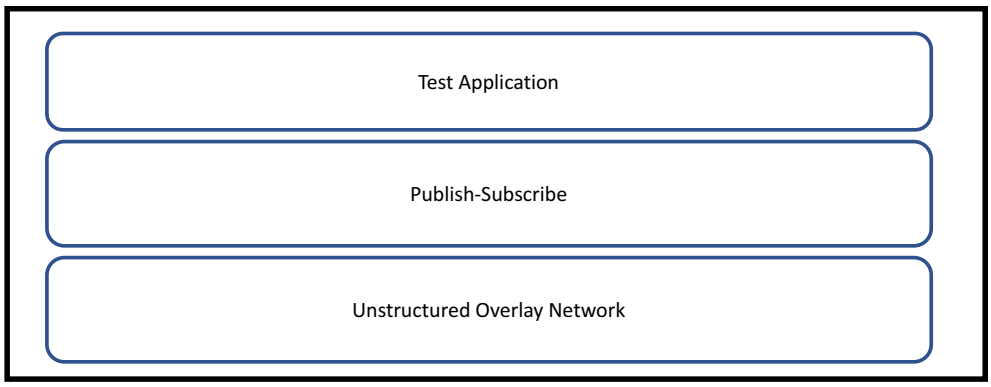
## Implementation Suggestions

Distributed Hash Table (DHT): You can implement either Chord or Pastry. You can make small adjustments to the overlay topology (i.e, add additional links) to make the dissemination process more efficient.

Publish Subscribe Layer: You should rely on the DHT interface to ensure that you store information about the topics of interest for each node at a single *elected* node for that topic. You should consider that faults might happen, but avoid to use replication. Instead, consider refreshing subscriptions periodically, such that if a failure occurs, only a few messages posted to some topics will be missed.

## Part II

# Option B: Publish-Subscribe using Unstructured Approaches




Representation of a Node in the System

In this option the students are asked to implement a publish-subscribe protocol on top of an unstructured overlay network (also known as unstructured overlay network).
As we will discuss in the lectures, an unstructured overlay network organizes nodes in a random topology that ensures global connectivity (i.e, there is at least on path in this logical network that allows a message to reach from any node to every other node in the system). The interface of this type of overlay is quite simple, it exposes an operation denoted, in the context of this project, as `neighbors( N )` that expose to the requesting protocol a list of, at most $N$, current neighbors of that node (i.e., a random sample).
We consider a publish subscribe system similar to the one in option A, which offers to other protocols the following interface:

`subscribe ( TOPIC )`: That notifies the system that the protocol above is interested in received (through a notification) all messages published that are tagged with `TOPIC`.

`unsubscribe ( TOPIC )`: That notifies the system that the protocol above is no longer interested in receiving messages tagged with `TOPIC`âĂę

`publish ( TOPIC, message )`: That publishes, to the system, a `message` tagged with `TOPIC`.

For simplicity, we consider that messages are tagged with exactly one topic.

## Implementation Suggestions

Partial View Membership: You can get ideas from Cyclon, Scamp, or HyParView. While you could implement one of these algorithms you can also make modifications to them, or combine ideas from multiple algorithms.

Publish Subscribe Layer: For the subscribe/unsubscribe operation you can limit this to a local operation (i.e, each process can simply store local information about the topics currently subscribed). To disseminate information, you can use a gossip-based broadcast protocol, and filter which messages are delivered to the application locally. You can also think about optimization that might allow to reduce the amount of messages being transmitted without impairing the delivery rate.

## Part III

# Option C: Individual Project

The general goal of the individual project is to build correct and provable abstractions on top of the Yggdrasil framework.
These abstractions should be possible to demonstrate in a large-scale setting (tens of devices) operating in the wireless ad hoc model (which is somewhat different – and a bit more challenging – than the network model usually considered in the context of ASD.
Abstractions build on this model must be experimentally accessed using simple demo applications that make use of the application in a stand alone way, while gathering relevant performance metrics during the execution. These demo applications **must** be able to operate without a human operator.
The report **must** include very concrete correctness arguments for the abstraction being developed (closer to a formal demonstration than a simple correctness argument).
Some of the abstractions that can be considered are:

Broadcast protocols.

Routing protocols.

Membership services.

Students aiming at doing this option **must** discuss and settle goals with the course professor before starting working on that.

## 2 Evaluation Rules & Criteria

The project includes both the delivery of the code and a written report that must contain clear and readable pseudo-code for each of the implemented layers alongside a description of the intuition of the protocol. A correctness argument for each layer will be positively considered during grading. The written report should provide information about any experimental work conducted by the students to evaluate their solution in practice (including results).
The project will be evaluated by the correctness of the implemented solutions, its efficiency, and the quality of the implementation. With a distribution of $50\%$, $25\%$, and $25\%$ respectively for each of the layers.
The quality and clearness of the report of the project will impact the final grade in $10\%$, however the students should notice that a poor report might have a significant impact on the evaluation of the correctness of the used solutions (which weight $50\%$ of the evaluation for each component of the solution).
Each component of the project will have a maximum grade of (out of 20):

Developed Distributed Protocols: 12/20

Test Application and Evaluation: 6/20.

Written Report: 3/20

*(Yes, the sum of this is not 20)*

## 3 Delivery Rules

Delivery of all components of the project is due on 18 October 2018 (Pending confirmation, some extra days might be provided after coordinating with the Course Coordination).
The methods for delivering the project will provided later.