NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS
Department of Physics and Informatics and Telecommunications

Master Thesis in Control and Computing

Control of communication with autonomous robotic devices through optimal stopping theory

Ioannis F. Galanis
Student ID: 2015503

Supervisor (or supervisors): **Stathes Hadjiefthymiades,** Associate Professor

ATHENS

FEBRUARY 2018

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΟΝ ΗΛΕΚΤΡΟΝΙΚΟ ΑΥΤΟΜΑΤΙΣΜΟ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Έλεγχος επικοινωνίας με αυτόνομες ρομποτικές συσκευές μέσω της θεωρίας της βέλτιστης παύσης

Ιωάννης Φ. Γαλάνης

Επιβλέπων (ή Επιβλέπουσα ή Επιβλέποντες): **Ευστάθιος Χατζηευθυμιάδης,** Αναπληρωτής Καθηγητής

ΑΘΗΝΑ

ΦΕΒΡΟΥΑΡΙΟΣ 2018

Master Thesis in Control and Computing

Control of communication with autonomous robotic devices through optimal stopping theory

Ioannis F. Galanis

S.N.: 2015503

SUPERVISOR: **Stathes Hadjiefthymiades,** Associate Professor

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Έλεγχος επικοινωνίας με αυτόνομες ρομποτικές διατάξεις μέσω θεωρίας βέλτιστης παύσης

Ιωάννης Φ. Γαλάνης

Α.Μ.: 2015503

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Ευστάθιος Χατζηευθυμιάδης,** Αναπληρωτής Καθηγητής

ABSTRACT

Nowadays, the domain of robotics experiences a significant growth. Until now, most robots were found in the form of industrial machines. In recent years, smaller scale robotic systems have appeared, like driverless cars, drones etc. In this thesis we focus our attention on Unmanned Vehicles intended for the air, sea and ground (UxV). Such devices are typically equipped with numerous sensors that detect contextual parameters from the broader environment, e.g., obstacles, temperature, etc. UxV reports its sensors findings (telemetry) to other systems, e.g., back-end systems, that further process the captured information. At the same time the UxV receives control inputs, such as navigation commands from other systems, e.g., commanding stations. We investigate a framework that monitors network condition parameters such as signal strength and prioritizes the transmission of control messages and telemetry. This framework relies on the theory of optimal stopping, which is concerned with the problem of choosing a time to take a particular acction in order to maximize an expexted reward or minimize an expected cost. We apply the optimal stopping theory to online assess the trade-off between the delivery of the messages and the network quality statistics and optimally schedule critical information delivery to back-end systems.

SUBJECT AREA:

KEYWORDS: Ad-hoc, sensor, mesh and vehicular wireless networks; Context-awareness in wireless, mobile and multimedia networks; Device-to-device Communications

ΠΕΡΙΛΗΨΗ

Στις μέρες μας, ο τομέας της ρομποτικής παρουσιάζει σημαντική ανάπτυξη. Μέχρι τώρα τα ρομποτικά συστήματα χρησιμοποιούνταν κυρίως σε βιομηχανικές μονάδες. Τα τελευταία χρόνια έχουν αρχίσει να εμφανίζονται και ρομποτικά συστήματα μικρότερης κλίμακας, όπως π.χ. αυτοκίνητα χωρίς οδηγό, μη επανδρωμένα αεροσκάφη κ.α.. Στη παρούσα διπλωματική εργασία εστιάζουμε την προσοχή μας στα μη επανδρωμένα οχήματα που προορίζονται για τον αέρα, τη θάλασσα και το έδαφος (Unmanned x Vehicle - UxV). Τέτοιες συσκευές είναι συνήθως εφοδιασμένες με πλήθος αισθητήρων οι οποίοι ανιχνεύουν συναφείς παραμέτρους από το ευρύτερο περιβάλλον, π.χ. εμπόδια, θερμοκρασία. Οι αισθητήρες μεταφέρουν τις πληροφορίες που συλλέγουν (τηλεμετρία) σε άλλα συστήματα, π.χ. μονάδες στήριξης, τα οποία επεξεργάζονται περαιτέρω τις συλλεγόμενες πληροφορίες ενώ το UxV λαμβάνει εντολές ελέγχου, όπως π.χ. εντολές πλοήγησης, από άλλα συστήματα, π.χ. σταθμούς ελέγχου. Εξετάζουμε ένα πλαίσιο που παρακολουθεί τις παραμέτρους της κατάστασης του δικτύου, όπως είναι η ισχύς του σήματος και δίνει προτεραιότητα στη μετάδοση μηνυμάτων ελέγχου και τηλεμετρίας. Αυτό το πλαίσιο βασίζεται στη θεωρία βέλτιστης παύσης, η οποία έχει ως στόχο την επίλυση του προβλήματος της επιλογής της χρονικής στιγμής κατά την οποία πρέπει να προβούμε σε μια συγκεκριμένη ενέργεια. Εφαρμόζουμε την θεωρία βέλτιστης παύσης ώστε να επιτύχουμε τον ιδανικό προγραμματισμό της παροχής πληροφοριών στα συστήματα στήριξης, μέσω της άμεσης αξιολόγησης των μηνυμάτων και των στατιστικών ποιότητας του δικτύου.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ:

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ:

ACKNOWLEDGMENTS

I would like to express my gratitude to Professor Stathes Hadjiefthymiades, my thesis supervisor, for his general advices and his suggestions concerning the way i should approach certain aspects of this research.

I would also like to extend my thanks to Dr. Kyriaki Panagidi for her guidance and encouragement. The door to Dr. Panagidi’s office was always open whenever i had a question about my research or writing.

Finally, i wish to thank my family for their continuous support throughout my studies.

Author

Ioannis Galanis

CONTENTS

LIST OF TABLES

List of Drawings

1. INTRODUCTION

Introduction

The rapid development of technology introduced the use of unmanned vehicles (UxVs- ‘x’ can stand for ‘a’ aerial, ‘g’ ground or ‘s’ sea vehicles) in areas of everyday life such as environmental monitoring, agricultural applications, disaster management, regardles of the domain that they belong to, i.e., aerial, ground or underwater. The characteristics that made them so popular are their degree of autonomy, i.e., the ability to make decisions without human intervention, the endurance and the payload they can support. In the same time, the paradigm of an intelligent network appeared which connects many types of devices, also named as “things”, to the Internet having as a goal the information exchange and communication. The vision of Internet of Things (IoT) has attracted the interest of researchers and practitioners who estimate that, by 2020, 25 billion devices, includint unmanned vehicles, will be connected to IoT networks worldwide.

UxVs have, as a goal, the successful execution of a mission. A mission is often described as a trajectory with specific way-points in which the vehicle is ordered to approach and gather various measurements from sensors or images from cameras. A Ground Control Station (GCS) is the terrestial system, which acts as a coordinator or master node at distance responsible for data acquisition and transmission. The communication between the unmanned vehicle and the GCS is established via wireless communications. A key feature of UxVs is the control of a possible mission. A mission is created by the users and, then, GCS is responsible for the successful execution of the mission autonomously. It is evidenced that it is necessary to have adequate mechanisms that permit connectivity, control and communications among the devices and especially in devices that move autonomously. GCS control messages shall be delivered with a high assurance of low or minimal time delay to enable real-time management, monitoring, control, and feedback loops. Therefore this feature is transformed to a challenge in IoT paradigm where the network structures and types are not fixed, wireless links cannot span large distances and the saturation of bandwidth and frequent disconnections can occur.

This thesis introduces a framework that monitors network condition through signal strength and prioritizes the transmission of control message and telemetry trying to rationally exploit the time varying resources that the network can offer. Essentially we are discussing about a time-multiplexed mechanism. The thesis aims to provide the insight of adopting the principles of the Optimal Stopping Theory (OST) to establish an *optimal stopping rule*, which provides the best time instance to maximize an expected pay off.

2. ROS Control decision for robots

Robot Operating System (ROS)[2] is an open-source, meta-operating system/framework which is of a wide use in the field of robotics. The concept behind the creation of such software is for it to be able to be used in other robots with as few and small code changes as possible. What can be gained by the use of this software is the capacity to create functionalities which can then be imported and applied effortlessly in other robots without the need for any radical changes.

Several ROS projects have been set up by numerous research institutions which have led to code samples being made available. In addition, commercial companies are already adapting their products so that they are compatible with ROS. As a standard practice, such platforms come with a large amount of available code including examples and simulation programs that allow developers to utilize them with as little difficulty as possible.

Sensors and actuators utilized in the field of robotics have also undergone changes so that they are compatible with ROS. The number of devices which can be used in this framework is being increased day by day. Standard operating system facilities such as hardware abstraction, low-level device control, implementation of commonly used functionalities, message passing between processes, and package management are provided in ROS. It is based on graph architecture with a centralized topology where processing takes place in nodes that may receive or post, such as multiplex sensor, control, state, planning, actuator, and so on. The library is geared towards a Unix-like system (Ubuntu Linux 16.04 LTS was used for the development of the code in this thesis).

Three conceptual levels make up the ROS architecture: the Filesystem level, the Computation Graph level and the Community level[2] .

2.1 Publish-subscribe pattern

In software architecture, publish-subscribe is a messaging pattern where the producers of messages, called publishers, do not send the messages directly to specific receivers, called subscribers, but instead categorize published messages into classes without knowledge of which subscribers, if any, there may be. Similarly, subscribers “subscribe” to receive one or more classes of messages that are of interest, without knowledge of which publishers, if any, there are.

This pattern has the benefit of greater network scalability and a more dynamic network topology, with the downside of a decreased flexibility to modify the publisher and the structure of the published data.

In the publish-subscribe model, subscribers typically receive only a segment of the total number of messages published. The process of selecting messages for reception and processing is called filtering which comes in two common forms: 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 in 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.

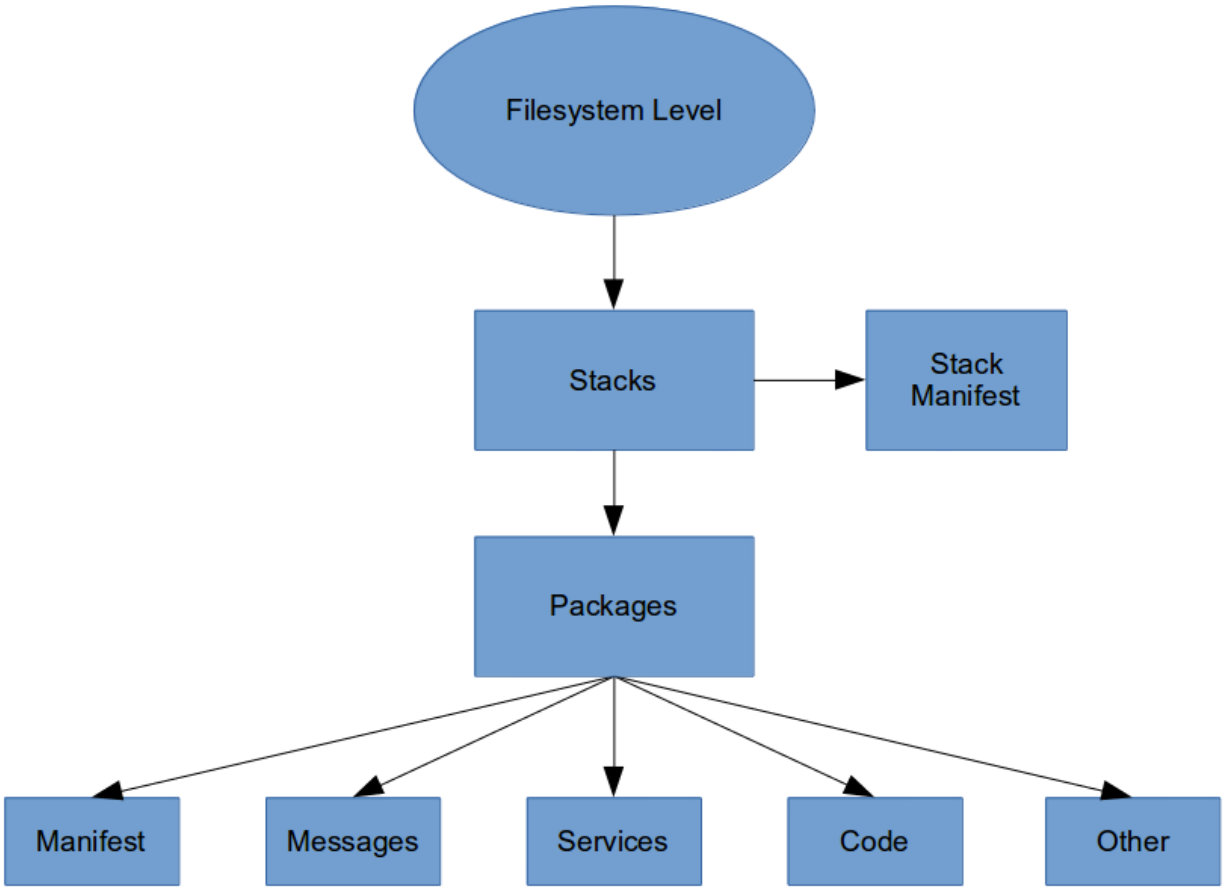
- In a content based system, messages are only delivered to a subscriber if the attributes or content of those messages match constraints defined by the subscriber. The subscriber is responsible for classifying the messages.

Some systems support a hybrid of the two, publishers post messages to a topic while subscribers register content-based subscriptions to one or more topics. In this thesis we will use only topic-based subscription systems. When we refer to a publish-subscribe system we will mean topic-based.

Publishers are loosely coupled to subscribers, and need not even know of their existence, With the topic being the focus, publishers and subscribers are allowed to remain ignorant of system topology. Each can continue to operate normally regardless of the other. In the traditional tightly coupled client-server paradigm, the client cannot post messages to the server while the server process is not running, nor can the server receive messages unless the client is running.

2.2 Filesystem Level

The first level, which is called the Filesystem level[2] includes a number of concepts that explain the internal structure, the folder structure and the basic files needed for it to operate.



Drawing 1: ROS filesystem organization

The folders that make up a ROS program include files describing their functionalities.

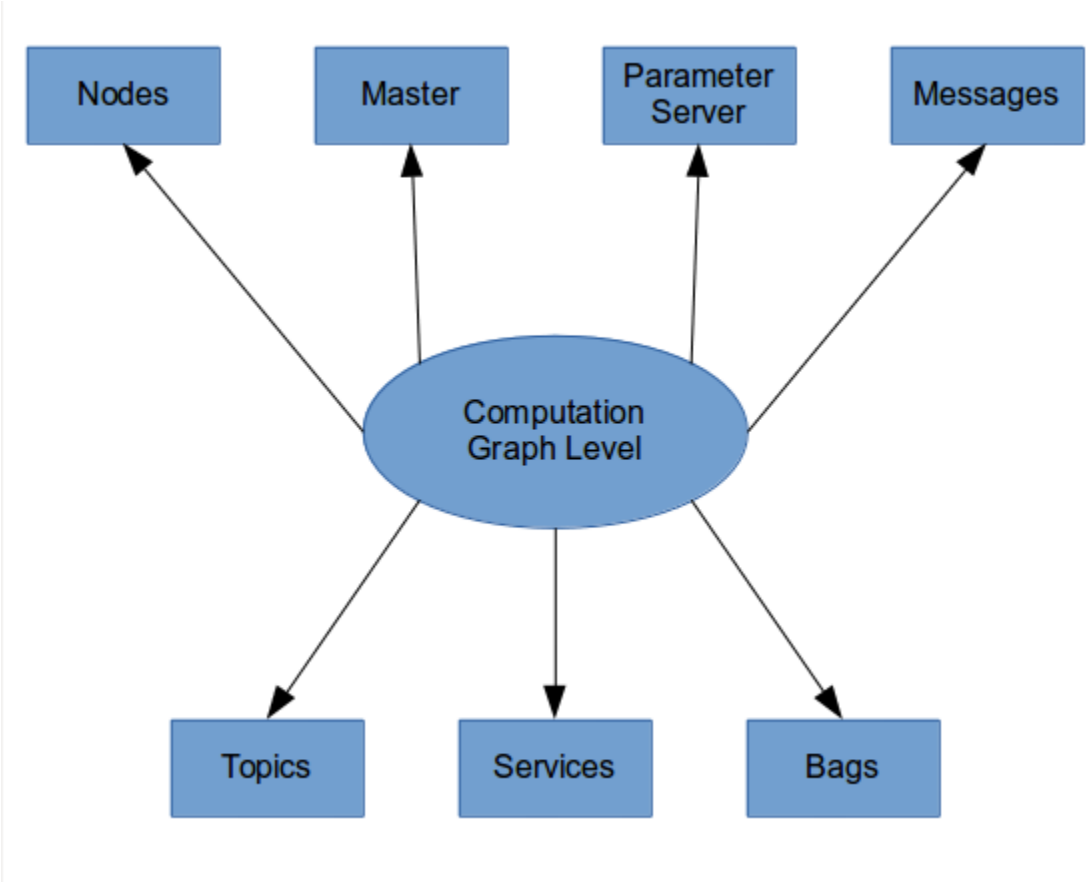
- Packages comprising the atomic level of ROS. The minimum structure and content that is necessary for the creation of a program within ROS is a package that may contain ROS runtime execution programs, which are called nodes, configuration files etc. The packages aim is to make the use of functionality easy in an efficiently and organized way that allows the software to be utilized in various projects. Thus, packages act as modular building blocks, a fact that drastically increases their reusability.
- Manifests are the medium providing metadata about a package which can include license information, dependencies, compiler flags etc. A file called manifests.xml is used for managing manifests.
- Message types define the structure of data for messages within ROS and service types.
- Service types provide the service descriptions which are stored in a (.srv) file. The request and response data structures to be used for services in ROS are also defined by the service types.
- Stack is made up of several packages featuring some functionality. There are numerous such stacks in ROS for various uses, such as the navigation stack.

- Stack manifests are responsible for the provision of data about a stack, including its license information and its dependencies on other stacks. A file called stack.xml is used for managing stack manifests.

2.3 Computation Graph Level

The second level which is called Computation Graph Level is where communication between processes and systems takes place. All the processes are connected to a network created by ROS. This network allows any node in the system to communicate with other nodes, access any information sent and transmit to the network.

The basic components in this level are nodes, the Master, the Parameter Server, messages, services, topics, and bags, all of which are used in various ways to provide data to the graph:



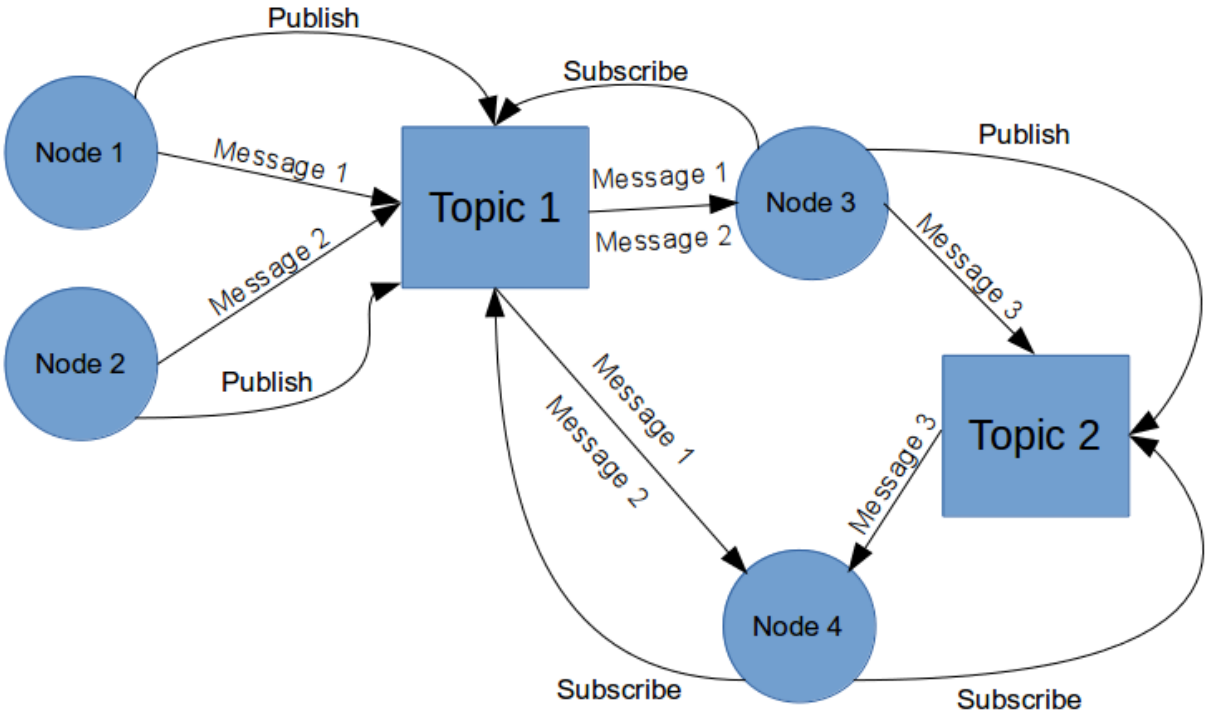
Drawing 2: ROS Computation Graph organization

2.3.1 Nodes

Nodes[2] are processes interacting with the ROS network. The name of each node in the system must be unique. Executing multiple copies of the same process is possible as long as a different node name is used for each copy which should be considered to be a separate node. The use of a different name prevents the communication between nodes from becoming ambiguous. A node can subscribe to topics to receive information, perform computation, control sensors and actuators, and publish data to topics for other nodes to use. Several nodes are usually executed by a system in order to control different functions. It is more efficient to utilize a large number of nodes each performing a single functionality than utilize a single large node that is responsible for multiple functionalities, as the former ensures increased fault tolerance and each functionality is created by a separate piece of code, thus simplifying the system. Nodes are created with the use of a ROS client library; in the case of java, rosjava is the library that is used.

2.3.2 Topics

Topics[2] are used so that nodes transmit data to each other. Topics are hubs characterized by a specific name which allow nodes to publish and subscribe without the need for direct interaction with each other. As a result, the production and consumption of data are not interdependent and a topic does not require the existence of a publishing node. The ROS message type by which each topic is strongly typed is the type of the messages that can be published to this topic. For a node requiring data to be able to subscribe to topics containing this data, it is necessary that the node has the same message type. It’s important that the name of the topic should be unique to avoid problems and confusion between topics of the same name.



Drawing 3: Example of message transmition between nodes and topics.

2.3.3 Messages

Messages are used to publish data to topics. Each message is structured in a simple way so that a standard type or one developed by the user is used. The standard ROS naming convention is used by message types, that is the name of the package, followed by /, and the name of the .msg file.

In ROS, you can find a lot of standard types to use in messages as shown in the following table:

Table 2.1: ROS message standard types for C++ and Python

Primitive type	Serialization	C++	Python
bool	Unsigned 8-bit int	uint8_t	bool
Int8	Signed 8-bit int	int8_t	int
uint8	Unsigned 8-bit int	uint8_t	int
int16	Signed 16-bit int	int16_t	int
uint16	Unsigned 16-bit int	uint16_t	int
int32	Signed 32-bit int	int32_t	int
uint32	Unsigned 32-bit int	uint32_t	int
int64	Signed 64-bit int	int64_t	long
uint64	Unsigned 64-bit int	uint64_t	long
float32	32-bit IEEE float	float	float
float64	64-bit IEEE float	double	float
string	ASCII string (4-bit)	std::string	string
time	Sec/nsecs signed 32-bit ints	ros::Time	rospy.Time
duration	Secs/nsecs signed 32-bit ints	ros::Duration	rospy.Duration

In ROS, the data values published by ROS nodes are described using a simplified description language [2] . By describing the data values in this way, it is possible to generate the appropriate source code for these types of messages in various programming languages. Although there are several predefined messages available in ROS, clients are offered the choice of developing a message type of their own and storing it in the msg/folder of their package. In this folder, the .msg files define the messages.

ROS has a lot messages predefined, but it is possible to develop a new message and store it in the msg/folder of your package. Inside that folder, some files with the .msg extension define the messages.

A message is made up of two essential parts: fields and constants. Fields is the part that defines the type of data that is transmitted in the message, such as int32, float32, string, or client created types. Constants is the part that defines the name of the fields.

An example of a message file is as follows:

- int32 id
- float32 vel
- string name

Header is a special ROS data type. It is used to add information, such as the timestamp, the frame etc. In this way messages can be numbered, making it possible to identify the node publishing the message. There is also the possibility of adding other functions visible to the user which are handled by ROS.

The header type contains the following fields:

- uint32 seq
- time stamp
- string frame_id

With the aid of Header, recording the timestamp and frame of what is happening with the robot is made possible.

In ROS there are certain tools used to work with messages. For example the tool rosmsg prints out the message definition information and can locate the source files that use a specific message type.

2.3.4 Services

Services[2] provide direct communication between nodes for request and response messages. Since no standard services are available for nodes it is necessary for the user to develop the services themselves. The srv folder is used to store the source code files. As with topics, there is a specific service type assigned to each service, which is the package resource name of the .srv file.

2.3.5 Bags

In an efficiently designed ROS system, a primary feature[3] is that the components that consume information are not aware of the method used to produce that information. Any time the required messages are published, a properly designed node functions without the other node or nodes publishing the messages being known to it. ROS creates files called bags in the .bag format so that all the information of the messages, topics, services etc can be stored. Processing, analyzing and visualizing the flow of messages can be done by using bags. The rosbag tool, which subscribes to one or more ROS topics and stores message data as they are received, is used to create bags. In ROS, it is possible for a bag file to be reproduced as a real time session in which the messages containing the same data will be published to the topics in the same time intervals. In this way we

can run the robot a number of times, record the messages published to the necessary topics and then have the messages to these topics replayed several times in a simulated environment while at the same time we experiment with the software processing those data.

2.3.6 Master

Master[2] is the domain name system server. Its role is to store the registration information of topics and services for ROS nodes. It tracks services, publishers and subscribers to topics. The purpose of the master is to make it possible for individual ROS nodes track each other so that communication between them is achieved in a peer-to-peer fashion. The parameter server is also provided by the ROS master.

2.3.7 Parameter Server

Parameter Server is a shared, multivariable dictionary to which access can be obtained via a network. It is used by nodes for the storage and retrieval of parameters at runtime. The ROS Parameter Server is implemented using XML-RPC (a remote procedure call protocol which uses XML to encode its calls and HTTP as a transport mechanism), which means that its API can be accessed via normal XML-RPC libraries. The Parameter Server uses XML-RPC data types for parameter values. These data types include the following:

- 32-bit integers
- Booleans
- Strings
- Doubles
- ISO 8601 dates
- Lists
- Base 64-encoded binary data

A user can also set a parameter from the command line by using:

rosparam set <param_name> <param_value>

2.4 ROS Community Level

The third level which is called the Community level is made up of ROS resources that allow the exchange of software and knowledge between separate communities.

2.5 Sensors

There are several types of sensors that can be found in a robot and are supported by official ROS packages and many more supported by the ROS community[4] .

1. 1D range finders
2. 2D range finders
3. 3D Sensors (range finders and RGB-D cameras)
4. Audio / Speech recognition
5. Cameras

6. Environmental (like measuring wind speed and direction)
7. Force / Torque / Touch Sensors
8. Motion Capture
9. Pose estimation (GPS / IMU[1])
10. Power Supply
11. RFID (Radio-frequency identification)[2]

2.6 ROS Commands

ROS provides users with a variety of tools in order to make navigation through the ROS filesystem and debugging as simple as possible. A few basic ROS commands are shown in Table 2.

Table 2.2: ROS commands

roscore	Starts ROS master
rosrun <pkg_name> <node_name>	Starts executable node
rosrun turtlesim turtlesim_node	Starts simple movement simulator
rosrun turtlesim turtle_teleop_key	Control the turtlebot movement with arrow keys
rospack list	Lists all installed ROS packages
rospack find <pkg_name>	Prints file path to package
rosls <pkg_name>	Lists all files in package directory

[1]IMU: Inertial measurement unit

[2]Radio-frequency identification (RFID) uses electromagnetic fields to automatically identify and track tags attached to objects.

roscd <pkg_name>	Allows you to change directories using a package name, stack name, or special location
rosnode info <node_name>	Display information about a node, including publications and subscriptions.
rosnode kill <node_name>	Kill one or more nodes by name. It is not guaranteed to succeed. If a node is hung or set to “respawn” in roslaunch, it may either fail to die or may quickly reappear.
Rosnode cleanup	Purge the registration of any node that cannot be contacted immediately. Prints list of unreachable nodes which has to be confirmed. It can potentially unregister functioning nodes.
rqt_graph	Provides a GUI plugin for visualizing the ROS computation graph.
rostopic echo </topic_name>	Prints topic messages to screen
Rostopic info </topic_name>	Provides data on topic such as type, subscribers and publishers.
rostopic list	Lists all active topics
rosbag record -O <filename></topic>	Starts rosbag tool to record data from a desired topic.

3. Optimal Stopping Theory

The theory of optimal stopping is concerned with the problem of choosing an optimal time to take a given action based on sequentially observed random variables in order to maximize an expected reward or minimize an expected cost. Problems of this type can be found in areas of statistics and operations research.

3.1 Definition of the problem

Stopping rule problems[5] are defined by two objects,

- a sequence of random variables, $X_1, X_2, \ldots$,whose joint distribution is assumed known, and
- a sequence of real-valued reward functions

$$y_0, y_1(x_1), y_2(x_1, x_2), \ldots, y_\infty(x_1, x_2, \ldots).$$

Given these two objects, the associated stopping rule problem may be described as follows. You may observe the sequence $X_1, X_2, \ldots$ for as long as you wish. For each $n = 1,2, \ldots$,after observing $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$, you may stop and receive the known reward $y_n(x_1,\ldots, x_n)$ (possibly negative), or you may continue and observe X_{n+1}. If you choose not to take any observations, you receive the constant amount, y_0. If you never stop, you receive$y_\infty(x_1, x_2, \ldots)$. (We shall allow the rewards to take the value $-\infty$; but we shall assume the rewards are uniformly bounded above by a random variable with finite expectation so that all the expectations below make sense.)

Your problem is to choose a time to stop to maximize the expected reward. You are allowed to use randomized decisions. That is, given that you reach stage *n* having observed $X_1 = x_1,\ldots, X_n = x_n$, you are to choose a probability of stopping that may depend on these observations. We denote this probability by $\varphi_n(x_1, \ldots, x_n)$. A (randomized) stopping rule consists of the sequence of these functions,

$$\Phi = (\varphi_0, \varphi_1(x_1), \varphi_2(x_1, x_2), \ldots),$$

where for all n an d$x_1, \ldots, x_n, 0 \leq \varphi_n(x_1, \ldots, x_n) \leq 1$. The stopping rule is said to be non-randomized if each $\varphi_n(x_1, \ldots, x_n)$is either 0 or 1.

Thus, φ_0 represents the probability that you take no observations at all. Given that you take the first observation and given that you observe $X_1 = x_1, \varphi_1(x_1)$represents the probability you stop after the first observation, and so on. The stopping rule, φ, and the sequence of observations, $X = (X_1, X_2, \ldots)$, determines the random time N at which stopping occurs, $0 \leq N \leq \infty$, where $N = \infty$if stopping never occurs. The probability mass function of N given $X = x = (x_1, x_2, \ldots)$is denoted by $\psi = (\psi_0, \psi_1, \psi_2, \ldots, \psi_\infty)$, where

$$\begin{aligned} \psi_n(x_1, \ldots, x_n) &= P(N = n|X = x) \text{for} n = 0,1,2, \ldots, \\ \psi_\infty(x_1, x_2, \ldots) &= P(N = \infty|X = x). \end{aligned}$$

This may be related to the stopping rule φ as follows:

$$\begin{aligned} \psi_0 &= \varphi_0 \\ \psi_1(x_1) &= (1 - \varphi_0)\varphi_1(x_1) \\ &\vdots \\ y_n(x_1) &= [\prod_1^{n-1}(1 - \varphi_j(x_1, \ldots, x_j))]\varphi_n(x_1, \ldots, x_n) \\ &\vdots \\ \psi_\infty(x_1, x_2, \ldots) &= 1 - \sum_0^\infty \psi_j(x_1, \ldots, x_j) \end{aligned}$$

$\psi_\infty(x_1, x_2, \ldots)$represents the probability of never stopping given all the observations.

Your problem, then, is to choose a stopping rule φ to maximize the expectated return, V(φ), defined as

$$V(\varphi) = E_{yN}(X_1, \dots, X_N) = E_{yN} \sum_{j=0}^{=\infty} \psi_j(X_1, \dots, X_N) y_j(X_1, \dots, X_j)$$

where the "$= \infty$" above the summation sign indicates that the summation is over values of j from 0 to ∞, including ∞. In terms of the random stopping time N, the stopping rule φ may be expressed as

$$\varphi_n(X_1, \dots, X_n) = P(N = n | N \geq n, X = x) \text{for} n = 0,1, \dots.$$

The notation used is that of Section 7.1 of Ferguson (1967).

3.1.1 Loss VS Reward

Often, the structure of the problem makes it more convenient to consider a loss or a cost rather than a reward. Although one may use the above structure by letting $y_n$denote the negative of the loss, clarity is gained in such cases by letting $y_n$denote the loss incurred by stopping at n, and considering the problem to be one of choosing a stopping rule to minimize V(φ).

3.1.2 Random Reward Sequences

For some applications, the reward sequence is more realistically described as a sequence of random variables $Y_0, Y_1, \dots, Y_\infty$whose joint distribution with the observations $X_1, X_2, \dots$is known. The actual value of $Y_n$may not be known at time n when the decision to stop or continue must be made. However, allowing returns to be random does not represent a gain in generality because, since the decision to stop at time n may depend on $X_1, \dots, X_n$, we may replace the sequence of random rewards $Y_n$by the sequence of reward functions $y_n(x_1, \dots, x_n)$for $n = 0,1, \dots, \infty$where

$$y_n(x_1, \dots, x_n) = E\{Y_n | X_1 = x_1, \dots, X_n = x_n\}.$$

Any stopping rule φ for the payoff sequence $Y_0, Y_1, \dots, Y_\infty$should give the same expected

3.2 The Secretary Problem

A commonly known application of the optimal stopping theory is known as the Secretary Problem[5] . The problem is usually described as the problem of a decision maker who is called to choose the best secretary among a finite number of applicants with the goal of picking the best applicant.

1. There is one position available.
2. There are n applicants for the position where n is a known finite number.
3. You can rank the applicants linearly from worst to best without ties.
4. The applicants are interviewed sequentially.
5. As each applicant is interviewed you must either reject the applicant and interview the next one or accept the applicant and end the decision problem.
6. You must make the decision to accept or reject each applicant using only the relative ranks of the applicants interviewed so far.

7. A rejected applicant cannot be recalled later.
8. The objective of the general solution is to maximize the probability of selecting the best applicant. This is the same as maximizing the expected payoff, with payoff defined to be 1 if you do select the best, 0 otherwise.

We place this problem into the guise of a stopping rule problem by identifying stopping with acceptance. We may take the observations to be the relative ranks, $X_1, X_2, \dots, X_n$, where $X_j$is the rank of the jth applicant among the first j applicants, rank 1 being best. By assumption 4, these random variables are independent and $X_j$has a uniform distribution over the integers from 1 to j. Thus, $X_1 \equiv 1, P(X_2 = 1) = P(X_2 = 2) = 1/2$, ect.

Note that an applicant should be accepted only if it is relatively best among those already observed. A relatively best applicant is called a candidate, so the jth applicant is a candidate if and only if $X_j = 1$. If we accept a candidate at stage j, the probability we win is the same as the probability that the best candidate overall appears among the first j applicants, namely j/n. Thus,

$$y_i(x_1, \dots, x_j)\begin{cases} j/n & \text{if applicant j is a candidate,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that $y_0 = 0$and that for $j \geq 1, y_j$depends only on x_j.

This basic problem has a remarkably simple solution which we find directly without the use of the optimal rule for finite horizon problems:

$$V_j^{(T)}(x_1, \dots, x_j) = max\{y_j(x_1, \dots, x_j), E(V_{j+1}^{(T)}(x_1, \dots, x_j, X_{j+1})|X_1 = x_1, \dots, X_j = x_j)\} \quad (1)$$

Let $W_j$denote the probability of win using an optimal rule among rules that pass up the first j applicants. Then $W_j \geq W_{j+1}$since the rule best among those that pass up the first j + 1 applicants is available among the rules that pass up only the first j applicants. It is optimal to stop with a candidate at stage j if $j/n \geq W_j$. This means that if it is optimal to stop with a candidate j, then it is optimal to stop with a candidate at j + 1, since $(j+1)/n > j/n \geq W_j \geq W_{j+1}$. Therefore, an optimal rule may be found among the rules of the following form, $N_r$for some $r \geq 1$:

N_r: Reject the first r-1 applicants and then accept the next relatively best applicant, if any.

Such a rule is called a threshold rule with threshold r. The probability of win using $N_r$is

$$\begin{aligned} P_r &= \sum_{k=r}^{n} P(k^{th}\text{applicant is best and is selected}) \\ &= \sum_{k=r}^{n} P(k^{th}\text{applicant is best})P(k^{th}\text{applicant is selected|it is best}) \\ &= \sum_{k=r}^{n} \frac{1}{n} P(\text{best of first k - 1 appears before stage r}) \\ &= \sum_{k=r}^{n} \frac{1}{n}\frac{r-1}{k-1} = \frac{r-1}{n}\sum_{k=r}^{n} \frac{1}{k-1} \end{aligned} \quad (2)$$

where $(r-1)/(r-1)$represents 1 if $r = 1$. The optimal $r_1$is the value of r that maximizes P_r. Since

$$\begin{gathered} P_{r+1} \leq P_r \text{if and only if} \\ \frac{r}{n}\sum_{r+1}^{n} \frac{1}{k-1} \leq \frac{r-1}{n}\sum_{r}^{n} \frac{1}{k-1} \text{if and only if} \\ \sum_{r+1}^{n} \frac{1}{k-1} \leq 1, \end{gathered} \quad (3)$$

we see that the optimal rule is to select the first candidate that appears among applicants from stage $r_1$on, where

$$r_1 = min\{r \geq 1 : \sum_{r+1}^{n} \frac{1}{k-1} \leq 1\} (4).$$

The following table is easily constructed.

n	=	1	2	3	4	5	6	7	8
r_1	=	1	1	2	2	3	3	3	4
P_{r1}	=	1.0	.500	.500	.458	.433	.428	.414	.410

Table 3.1: Example of secretary problem probabilities

It is of interest to compute the approximate values of the optimal $r_1$and the optimal $P_{r1}$for large n. Since $\sum_{r+1}^{n} \frac{1}{k-1}$~$\log(\frac{n}{r})$, we have approximately $\log(n/r_1) = 1$, or $r_1/n = e^{-1}$. Hence, for large n it is approximately optimal to pass up a proportion $e^{-1}$of the applicants and then select the next candidate. The probability of obtaining the best applicant is then approximately e^{-1}.

3.2.1 The Parking Problem (Mac Queen and Miller (1960))

You are driving along an infinite street towards your destination, the theater. There are parking places along the street but most of them are taken. You want to park as close to the theater as possible without turning around. If you see an empty parking place at a distance d before the theater, should you take it?

Here, we model this problem in a discrete setting. We assume that we start at the origin and that there are parking places at all integer points of the real line. Let $X_0, X_1, X_2, \ldots$be independent Bernoulli random variables with common probability p of success, where $X_j = 1$means that parking place j is filled and $X_j = 0$means that it is available. Let if you do you lose |T-j|. You cannot see parking place j+1 when you are at j, and if you once pass up a parking place you cannot return to it. If you ever reach T, you should choose the next available parking place. If Y is filled when you reach it, your expected loss is then $(1-p) + 2p(1-p) + 3p^2(1-p) + \cdots = 1/(1-p)$,so that we may consider this as a stopping rule problem with finite horizon T and with loss

$$y_T = 0 if X_T = 0 \text{and} y_T = 1/(1-p)\ if X_T = 1 (5)$$

and for $j = 0, \ldots, T-1$,

$$y_j = T - j \text{if} X_j = 0 \text{and} y_j = \infty if X_j = 1. (6)$$

The value $y_j = \infty$forces you to continue if you reach a parking place j and it is filled.

We seek a stopping rule, $N \leq T$, to minimize EY_N.

First we show that if it is optimal to stop at stage j when $X_j = 0$, then it is optimal to stop at stage $j+1$when $X_{j+1} = 0$. As in Moser's problem, $V_j^{(T)}$depends only on X_j,and $X_j = 1$is a constant that depends only on $n-j$. It is optimal to stop at stage $n-j$if $y_{n-j} \leq A_j$. We are to show that if $n-j \leq A_j$, then $n-j-1 \leq A_{j-1}$. This follows from the inequalities, $n-j-1 < n-j \leq A_j \leq A_{j-1}$.

Thus, there is an optimal rule of the threshold form, $N_r$for some $r \geq 0$: continue until r places from the destination and park at the first available place from then on. Let $P_r$denote

the expected cost using this rule. Then, $P_0 = p/(1-p)$and for $r \geq 1$,$P_r = (1-p)r + pP_{r-1}$. We will show by induction that

$$P_r = r + 1 + \frac{2p^{r+1}-1}{1-p} \quad (7)$$

$P_0 = p(1-p) = 1 + (2p-1)/(1-p)$, so it is true for $r = 0$. Suppose it is true for $r-1$; then $P_r = (1-p)r + pP_{r-1} = (1-p)r + pr + p\,(2p^r - 1)/(1-p) = (r+1) + (2p^{r+1} - 1)/(1-p)$, as was to be shown. To find the value of r that minimizes (7), look at the differences, $P_{r+1} - P_r = 1 + (2p^{r+2} - 2p^{r+1})/(1-p) = 1 - 2p^{r+1}$. Since this is increasing in r, the optimal value is the first r for which this difference is nonnegative, namely, $min\{r \geq 0: p^{r+1} \leq 1/2\}$. For example, if $p \leq 1/2$, you should start looking for a parking place $r = 6$places before the destination.

3.3 Detecting a Change Point

Let us assume that we monitor a sequence of i.i.d. random variable $X_1, X_2,$ ...with a known distribution F_0. At some point T in time, unknown to you, the distribution will change to sime other known distribution F_1, and we have to sound an alarm as soon as possible after the change occurs. It is assumed that you know the distribution of T. If the cost of stopping after the change has occurred is the time since the change, and if the cost of false alarm that is of stopping before the chane has occurred is taken to be a constant c>0 then the total cost may be represented by

$Y_n = cI\{n < T\} + (n-T)I\{n \geq T\}$for n=0,1.., and $Y_\infty = \infty$

In this display I(A) represents the indicator function of a set: so for example $I\{n < T\}$is equal to 1 if n<T and to zero otherwise. Since T is a ranfom unobservable quantity we may replace $Y_n$by conditional expected value given $X_{1,\dots}, X_n$

$y_n = cP(T > n \mid F_n) + E((n-T)^{(+.)} \mid F_n)$for n=0,1.., and $Y_\infty = \infty$

Change-point detection has its origins almost sixty years ago in the work of Page[6] , Shiryayev[7] , and Lorden[8], who focused on sequential detection of a change-point in an observed stochastic process. The stochastic process was typically a model for the measured quality of a continuous production process, and the change-point indicated a deterioration in quality that must be detected and corrected. In our case, we are adopting this methodology for finding the best time instance for the controller strategy in order to maximize the possibility of successful message delivery by observing runtime network statistics.

4. Definition of Problem

4.1 Contribution

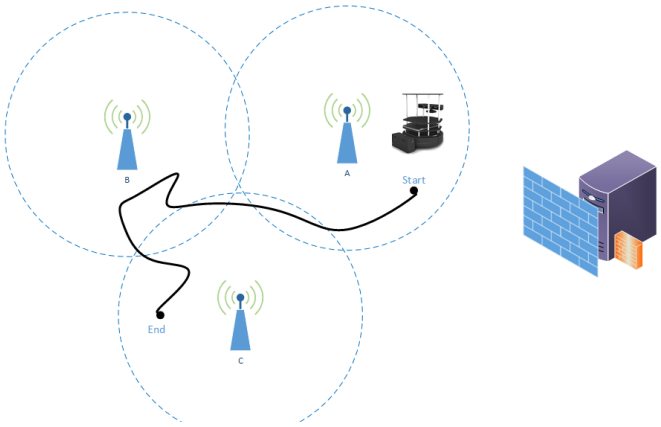
We assume that a user creates an outdoor mission for an unmanned vehicle. The vehicle is autonomous and not human commanded. Two components exist: a ground control station (GCS) and an unmanned ground vehicle (UGV). GCS sends commands, e.g., “Go-to” a point, “Pause” on a specific point, or “Abort” the mission and return *home*. UGV sends sensor value streams, e.g., temperature, humidity, images and its GPS position. Both GCS and UGV have as a goal the successful completion of the mission. Both components monitor the quality of the network. Quality of the network has high importance for the mission because significant commands or sensor values can be lost. The quality of the network can be discriminated as proposed in [1] . We can assume that even if control feedback is in high priority, telemetry can be paused for short time in need of crisis. When the quality of network changes from one state $S_1$to a state S_2then the strategy of GCS and UGV changes accordingly, i.e., UGV/GCS can pause the transmission of a command in order not to overload a saturated network, or to risk to loose completely the messages. The time in which a network change occurs and the period that one strategy $S_i$lasts are automatically computed in this framework. Thus, we propose a model of dynamic decision making adaptive to changes in network conditions by dynamically adjusting the transmission of control messages and telemetry based on an OST rule. Specifically, we contribute to:

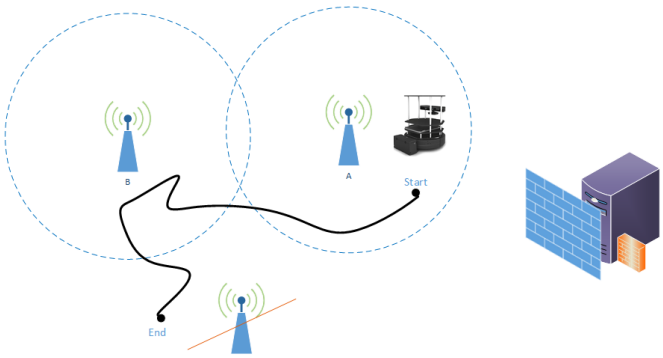
1) how we characterize the network conditions,
2) we define an optimal stopping rule for the discussed problem and how it is applied in our context,

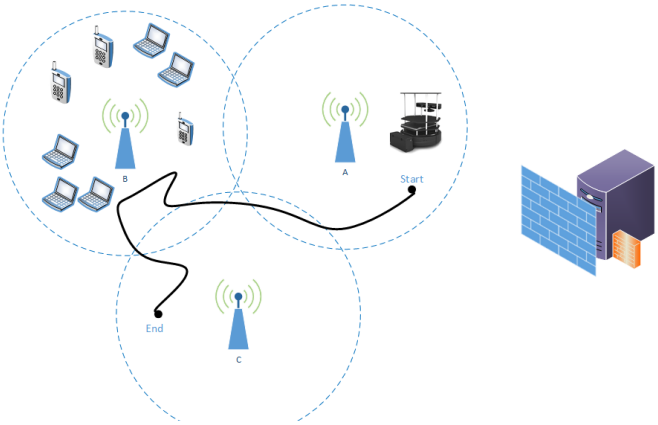
3) we provide a comprehensive performance evaluation of the proposed time-optimized scheme.

4.2 Problem Formulation

Let us assume an outdoor space in which a ground vehicle - UGV operates. In this space the controller initiates a mission. GCS sends commands to UGV, and the UGV sends back its position and sensor data streams. As UGV is moving in the space and communicating with the controller via a wireless connection, it may experience variations in signal based on the distance between its position and the access point, or an overloaded bandwidth. These can be defined as changes to network state. Therefore, the controller shall change its strategy by pausing the commands as long as the possibility of successful delivery of messages is considerably low. In our strategy, the point at which the controller changes from one strategy to the other based on the network quality statistics casts as an optimal stopping time problem, a.k.a. *change point detection*. For example a UxV can be engaged in a mission probably and can be tasked to explore places with full connectivity, with communication “holes”, or with limited or saturated bandwidth as shown in Drawings 4,5 and 6. The main question in these situations is how we timely *identify* a change to the network and for how long the transitioning from one state to another lasts.







4.3 Our Approach

Let us assume that $X_1, X_2, \dots, X_n$are independent and identically distributed random variables and are observed sequentially in real time. Consider also that there are two known probability density functions $f_0$and f_1, where $f_0 \neq f_1$. We are intrested in finding the *stopping time* that will detect a change from one distribution to another with the minimum delay based only on the realization of the random values $x_1, x_2, \dots, x_n$. We allow n to the state to the infinity denoting by this case that no change occurs. Let $\mathcal{F}_n, n \geq 1$be the σ-algebra generated by random variables (realizations) $\{X_1, X_2, \dots, X_n\}$and $\mathcal{P}_i$equal to the uncertainty class of distribution f_i, where i denotes the i-th distribution. A sequential change point detection procedure is characterized by a stopping time τ with respect to the observation sequence. The design of the quickest change-point detection procedure involves optimizing the trade off between two performance metrics: (i) the detection delay and (ii) the frequency flase alarm, as denoted in [9] . A false alarm is a wrongly detection of a change from one distribution to the other. A false positive rate is calculated as the ratio between the number of negative events wrongly categorized as network changes. In the minimax formulation of Lorden [8] , the change point is assumed to be an unknown deterministic quantity. The worst case detection delay is defined as:

$$D_n(t) = \sup_{n \geq 1} ess\sup \mathbb{E}_k[(\tau - k + 1)_+ | \mathcal{F}_{k-1}] \quad (8)$$

where $x^+ = max\{x, 0\}$and the False Alarm Rate (FAR) is defined in [9] as:

$$FAR(\tau) = \frac{1}{\mathbb{E}_\infty[\tau]}. \quad (9)$$

Here $\mathbb{E}_\infty[\tau]$defines the expected time between false alarms. Under the Lorden criterion the objective is to find the stopping rule that minimizes the worst-case delay subject to an upper bound on the false alarm rate:

$$min. D(\tau) s.t. Far(\tau) \leq \alpha. \quad (10)$$

The optimal solution to (10) was proved by [10] that it is given by the Cumulative Sum (CUMSUM) test proposed by Page[6] . Hence the optimal stopping time is given by:

$$\tau = inf\{n \geq 1, \max_{1 \leq k \leq n} \sum_{i=k}^{n} \quad L(X_i) \geq \eta\} \quad (11)$$

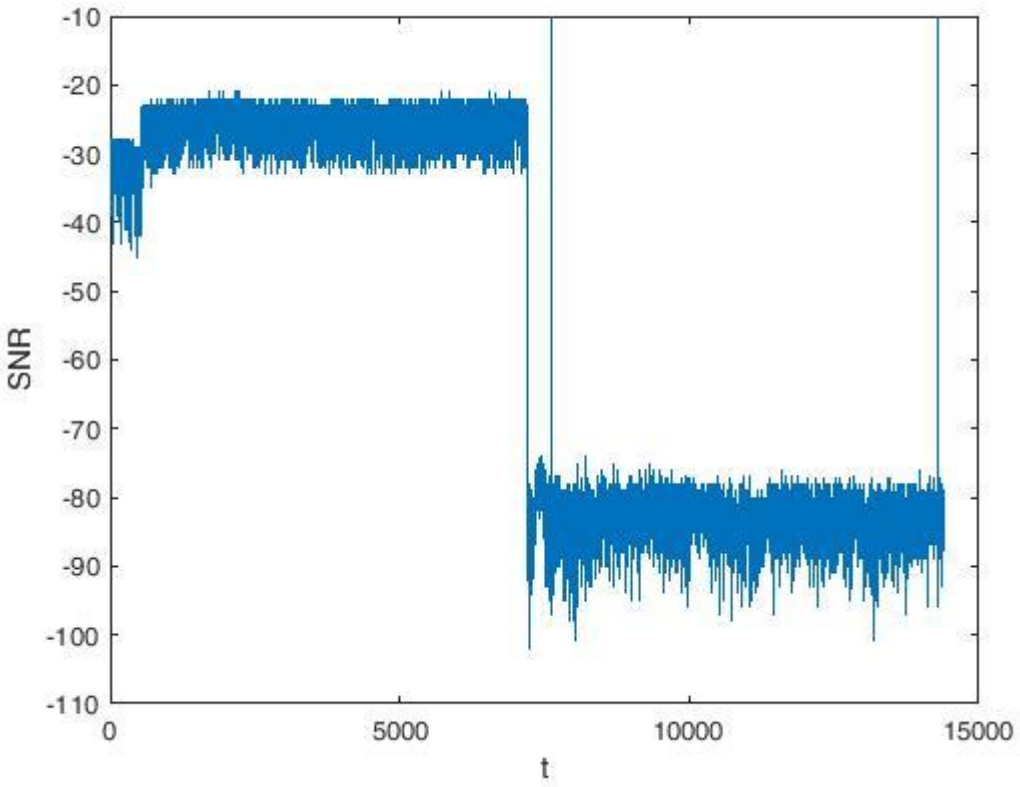
Let L denote the log-likelihood ratio between $f_0(X)$and $f_1(X)$defined as the Radon-Nikodym derivative, i.e., $L(X) = \log \frac{f_0(X)}{f_1(X)}$.The threshold $\eta$is chosen so that $\mathbb{E}_\infty[\tau] = \frac{1}{a}$.

Theorem 1: It is assumed that the following conditions hold

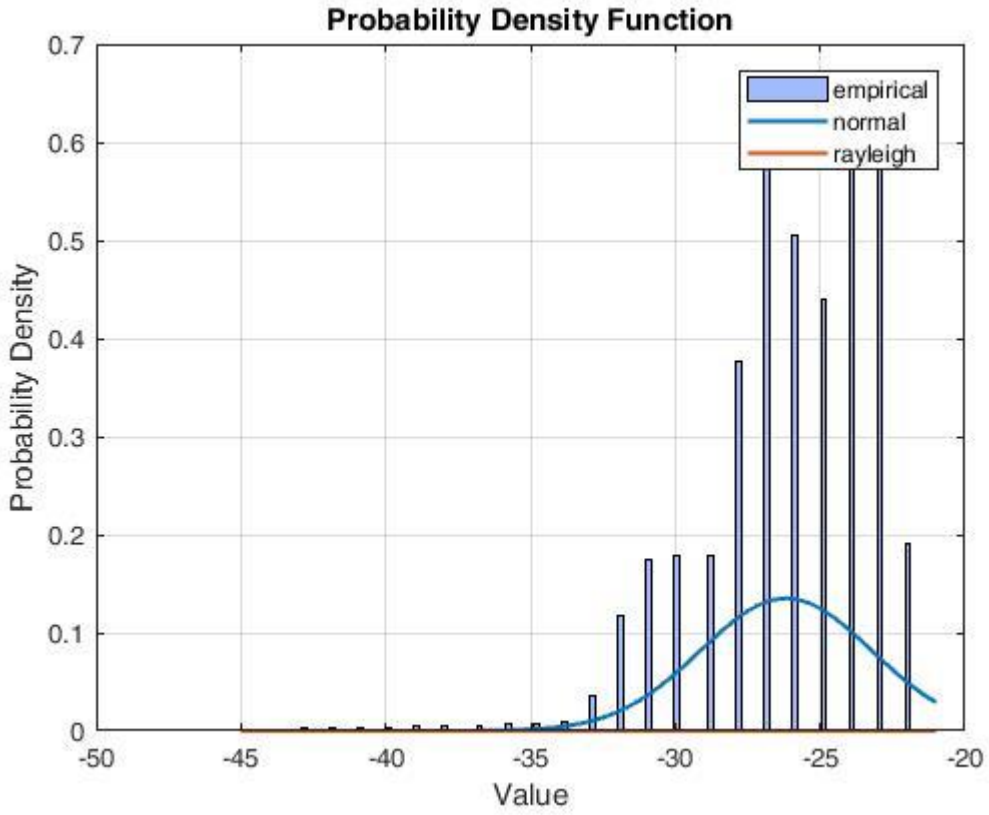
- Uncertainty classes of distributions $\mathbb{P}_0, \mathbb{P}_1$are jointly stochastically bounded by$(\overline{f_0}, \overline{f_1})$.
- All distributions $f_1 \in \mathbb{P}_1$are absolutely continuous with respect to $\overline{f_1}$,i.e.

$$f_1 \ll \overline{f_1}, f_1 \in \mathbb{P}_1 (12)$$

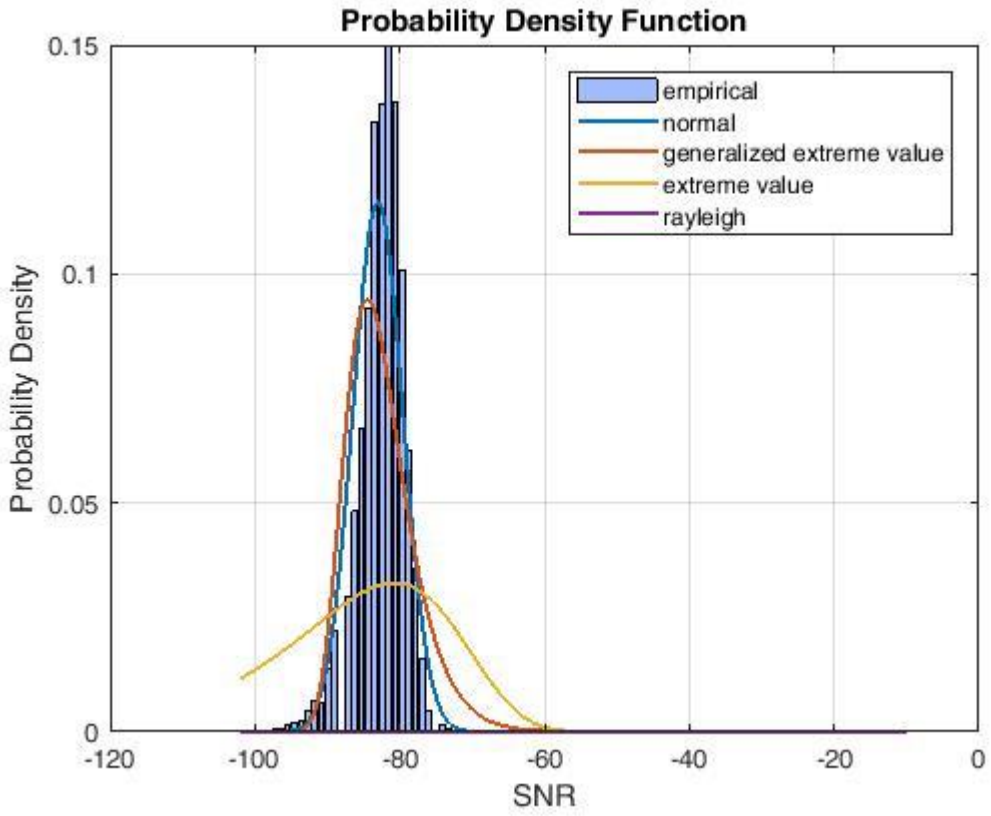
- Function $L^*(.)$representing the log-likelihood ration between $\overline{f_0}$and $\overline{f_1}$is continuous over the support of $\overline{f_1}$.



Drawing 7: SNR change from good to bad state.



Drawing 8: Probability density function f0 model fitting for good quality of SNR values with μ=-26.1728 and σ=2.95569.



Drawing 9: Probability density function f1 model fitting for bad quality of SNR values with μ=-83.021 and σ=3.44657.

The optimal stopping rule which is derived by [10] applying the CUMSUM test is the following:

$$\tau^* = inf\{n \geq 1: \max_{1\leq k\leq n} \sum_{i=k}^{n} \quad L^*(X_i) \geq \eta\} \text{(13)}$$

where threshold $\eta$is chosen so that $\mathbb{E}_{\infty}^{*}(\tau^*) = \frac{1}{a}$.

Proof: Let us assume the conditions in Theorem 1. Since the CUMSUM test is optimal for known distributions it is clear that the test given in (13) is optimal when the pre- and post-change distributions are $\overline{f_0}$and $\overline{f_1}$respectively. Therefore it would be enough to show that the values of $D(\tau^*)$and $FAR(\tau^*)$are obtained under any $f_0 \in \mathbb{P}_0$and any $f_1 \in \mathbb{P}_1$. $Y_i^*$is defined as the random variable $L^*(X_i)$where the pre-change and post-change distributions of the observations from the sequence $X_1, X_2, \dots, X_n$are $\overline{f_0}$and $\overline{f_1}$respectively. It is defined $Y_i$as the random variable $L^*(X_i)$when the pre- and post- change distributions are known and the observations are referring to SNR measurements, i.e., each time instance k the controller measures the SNR variableX_k.These measurements are stochastically drawn under the distribution$f_0$with mean$\mu_0$and varianceσ_0^2.Their distribution changes to a distribution$f_1$with mean value$\mu_1$and variance σ_1^2at some unknown point m so that SNR variables$X_k \sim f_0$for$k \leq m-1$and$X_k \sim f_1$for$K \geq m$,as shown in Drawing 7.

To estimate the two probability density functions $f_0$andf_1,a probability density function comparison method was adopted to derive the closest distributions to our SNR values. The method returns the model fitting of all the parametric probability distributions to the SNR and compare them graphically as shown in Drawing 8 for $f_0$and Drawing 9 for f_1.Based on these observations, we model$f_0$and$f_1$through the Normal distribution. Specifically, given the normal distributions $f_0$and f_1,the lo-likelihood ratio $L^*(X_i)$is then given as:

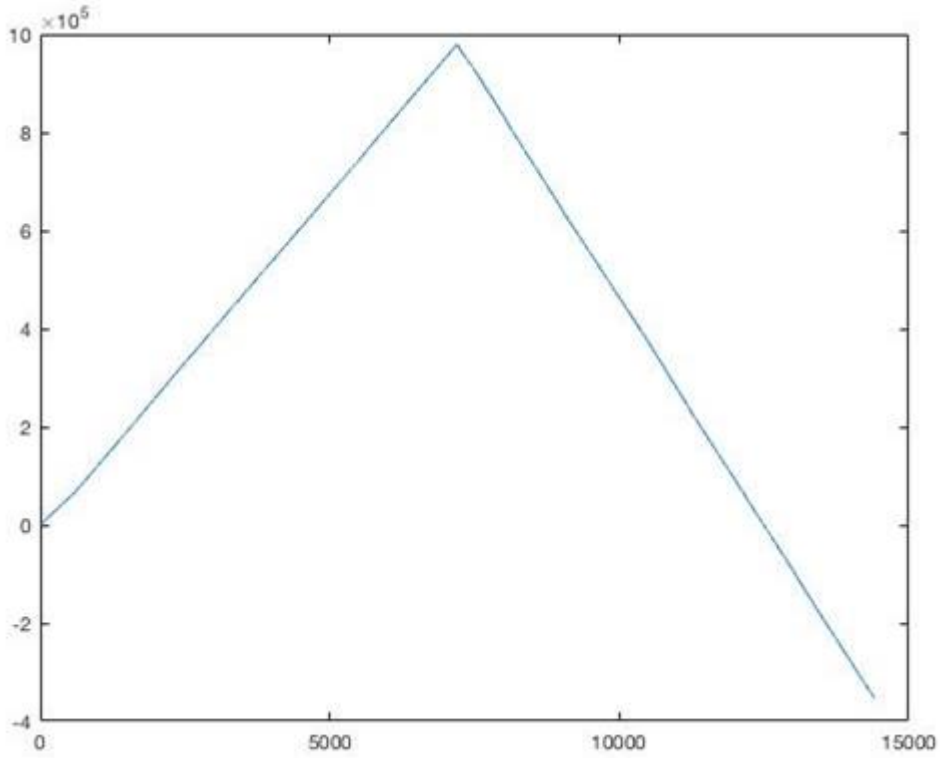
$$L^*(X_i; \mu_0, \sigma_0, \mu_1, \sigma_1) = \ln\frac{\sigma_1^2}{\sigma_0^2} + \frac{(X_i-\mu_1)^2}{2\sigma_1^2} - \frac{(X_i-\mu_0)^2}{2\sigma_0^2} \text{(14)}$$

It is evident that the FAR obtained by using the stopping rule $\tau^*$is independent of the true value of the post-change distribution. Let us assume that the change point is set to a timestep λ.Hence we must prove that for all $\lambda \geq 1$

$$\mathbb{E}_{\lambda}^{\infty}[(\tau^* - \lambda + 1)^1 | \mathcal{F}_{\lambda-1}] \succ \mathbb{E}_{\lambda}[(\tau^* - \lambda + 1)^1 | \mathcal{F}_{\lambda-1}], \text{(15)}$$

which establishes that the value of $D(\tau^*)$obtained under any $f_1 \in \mathbb{P}_1$is no higher than the value when the true post-change distribution is$\overline{f_1}$. The $A \succ B$operator denotes that A is jointly stochastically bounded by B. We can show, without the loss of generality, that for all i<λ, $Y_i$holds with probability 1. Under this assumption it is sufficient to prove that:

$$P_{\lambda}^{\infty}((\tau^* - \lambda + 1)^+ \leq N | \mathcal{F}_{\lambda-1}) \leq P_{\lambda}((\tau^* - \lambda + 1)^+ \leq N | \mathcal{F}_{\lambda-1}) \text{(16)}$$



Drawing 10: The behavior of the cumulative log-likelihood ratio corresponding to a change.

which would then establish (15). Since $\tau^*$is a stopping time, the event $(\tau^* - \lambda + 1)^+ \leq$ 0is$\mathcal{F}_{\lambda-1}$- measurable. Therefore with probability 1 (16) holds with equality for N=0. For $N \geq 1$we know that under stochastic ordering condition on $\mathcal{P}_1$it is true that:

$$Y_i \succ Y_i^{\infty}, \forall i \geq \lambda. (17)$$

Hence, between a change, the following equation holds:

$$\begin{aligned} \{\tau^* \leq N\} &= \{max_{1\leq n\leq N} max_{1\leq k\leq n} \sum_{i=k}^{n} L^*(X_i) \geq \eta\} \quad (10) \\ &= \{max_{1\leq k\leq n\leq N} \sum_{i=k}^{n} L^*(X_i) \geq \eta\}. \quad (11) \end{aligned} \quad (18)$$

It is easy to see that the function $N \geq 1$is a nondecreasing function for all the$i < \lambda$as we can experimmentally also see in Drawing 10.

Hence for $N \geq 1$the following holds:

$$\begin{aligned} P_{\lambda}^{*}((t^* - \lambda - 1)^+ &\leq N|\mathcal{F}_{\lambda-1}) \\ &= P_{\lambda}^{\infty}(\tau^* \leq N + \lambda - 1|\mathcal{F}_{\lambda-1}) \\ &= P_{\lambda}((Y_1^*, \dots, Y_{N+\lambda-1}^*) \geq \eta|\mathcal{F}_{\lambda-1}) \\ &\leq P_{\lambda}((Y_1, \dots, Y_{N+\lambda-1}) \geq \eta|\mathcal{F}_{\lambda-1}) \\ &= P_{\lambda}(\tau^* \leq N|\mathcal{F}_{\lambda-1}) \\ &= P_{\lambda}((\tau^* - \lambda + 1)^+ \leq N|\mathcal{F}_{\lambda-1}) \end{aligned} \quad (19)$$

This denotes that the equation $D(\tau^*)$(10) under any pair of distribution $(f_0, f_1) \in P_0 \times P_1$is no larger than $(\overline{f_0}, \overline{f_1})$and thus $\tau^*$is the optimal solution.

Drawing 11: Dataflow between the Turtle bot and the Ground Control Station

4.4 Code

The code makes heavy use of loose coupling and programming to interfaces and not implementations. This results in modularity which allows easy future changes of the algorithms used, whithout changing the rest of the code. The code is organized in the following packages.

4.4.1 Package: ost

4.4.1.1 Class: OST

The OST class has methods for the math of the optimal stopping algorithm. It has the following variables:

- private Double m0: the mean or expectation of the good quality distribution.
- private Double m1: the mean or expectation of the bad quality distribution.
- private Double s0: the standard deviation of the good quality distribution.
- private Double s1: the standard deviation of the bad quality distribution.

The following methods are implemented:

- public double f0(int x): returns the relative likelihood that the value x is equal to a random variable that follows the good distribution.
- public double f1(int x): returns the relative likelihood that the value x is equal to a random variable that follows the bad distribution.

- public static int minIndex(List<Double> list): returns the index of the minimum double number in the list.
- public static int maxIndex(List<Double> list): returns the index of the maximum double number in the list.

4.4.2 Package: logic

4.4.2.1 Class: DecisionMaker

DecisionMaker is the class which is responsible to characterize the quality of the signal, stop the producers when needed and start them when the quality is good enough. We will use one DecisionMaker for each group of producers that we require to follow the same rules. The algorithm that is used for the evaluation of the quality will be provided to the DecisionMaker from another class which will implement the DMLogic interface so it can be different between groups of producers and if we need to make changes in order to improve the algorithm the rest of the code wont be impacted. It has the following attributes:

- ArrayList<MyProducer> producers: A list of the producers the DecisionMaker is controlling.
- enum quality{goodQuality, badQuality}: used to store the current quality which can take either the goodQuality value if the wifi connection is evaluated to be good or badQuality if the connection is bad.

4.4.2.2 Interface: DMLogic

This interface is implemented by the classes that provide the algorithms which give send permission to the producers. It has a single method, getSendPermission(int q), which takes as an argument the current quality and returns a boolean that allows (true) to send the messages or denies (false) to send them.

4.4.2.3 Class: DMLogicOST

This class is implementing the DMLogic interface and is responsible for the optimal stopping algorithm. It has the following variables:

- OST ost: as described above the OST class provides us with methods that are used from the optimal stopping algorithm.
- int changeIndex: keeps track of the index where the last change happened so that the algorithm can start from the point of the change
- ArrayList<Double> Z: a list that is used to store the $\log(\frac{f_0(x)}{f_1(x)})$for each new value of x.
- ArrayList<Double> OS: a list that is used to store the $\sum_{i=changeIndex}^{n} \log(\frac{f_0(x_i)}{f_1(x)})$values.
- ArrayList<Integer> Change: a list that is used to store integers that represent either being in a good or bad quality state.

- double sum: its used to store the current value of $\sum_{i=changeIndex}^{n} \log(\frac{f_0(x_i)}{f_1(x)})$.
- double b: the sensitivity of the algorithm
- boolean sendPermission: this variable is set to true if the wifi quality is good and it switches to false when the wifi quality drops enough.
- int i: counts the current iteration of the algorithm.

The only method in this class is the getSendPermission(int q). It takes the current wifi quality as input and returns a boolean which can be used to either allow or block message transmission.

Table 4.1: getSendPermission method code

```
public boolean getSendPermission(int q){
     int C_INCREASE=10000; //rate of growth
     int C_DECREASE=-10000; //rate of reduction - when the quality
follows the bad distribution

     int result,OS_size;
     double q0,q1,t;

     q0=ost.f0 (q);
     q1=ost.f1(q);
     Z.add(Math.log((q0/q1)));
     sum+=Math.log((q0/q1));
     OS.add(sum);
     result=OST.minIndex(OS);
     OS_size=OS.size();

     if(i>1){
          if(Math.abs(OS.get(OS.size()-
1))<(Math.abs(OS.get(OS.size()-2))+b)){
               Change.add(C_INCREASE);
               result=OST.maxIndex(OS.subList(changeIndex,
OS_size));
               t=sum-OS.get(result);

               }else{
                    Change.add(C_DECREASE);
                    result=OST.minIndex(OS.subList(changeIndex,
OS_size));
                    t=sum-OS.get(result);
```

```
                }
                if(Change.size()-changeIndex>2) {
                      int numb1,numb2;
                      numb1=Change.get(Change.size()-2);
                      numb2=Change.get(Change.size()-1);
                      if(numb1!=numb2){// Change occured
                            sum=0.0;
                            changeIndex=i;
                            sendPermission = !sendPermission;
                      }
                }
      }
      i++;
      return sendPermission;
}
```

4.4.3 Package: producer_consumer

4.4.3.1 Class: OstProducer

OstProducer is the “producer” class responsible for the production of the messages. We use this class to create data messages and control messages. Each type of message will have its own instance of the producer class and will share the same DecisionMaker with the producers used for messages following the same priority rules. It has the following attributes

- String serverAddress: the address of the kafka server used for the exchange of the messages between the robot and the base.
- String keySerializer: the name of the serializer class utilized to convert the key of the messages to the correct format before sending them to the kafka server.
- String valueSerializer: the name of the serializer class utilized to convert the value of the messages to the correct format before sending them to the kafka server.
- String topicName: the name of the topic on the kafka server where the producer will publish its messages.
- private volatile boolean on: while this flag is set to true the producer continues sending messages, if set to false the producer pauses and waits untill it is set to true again.

4.4.3.2 Class: OstConsumer

OstConsumer is the “consumer” class responsible for the processing of the messages. We use this class to receive data messages and control messages. Each type of message will have its own instance of the consumer class. It has the following attributes:

- String topicName: the name of the topic to which this where this consumer will subscribe to receive its messages.
- String serverAddress: the address of the kafka server used for the exchange of the messages between the robot and the base.
- String keyDeserializer: the name of the deserializer class used to convert the key of the messages in a readable format.
- String valueDeserializer: the name of the deserializer class used to convert the value of the messages in a readable format.

4.4.4 Package: Simulation

4.4.4.1 Class: ExponentialFreqGenerator

This class is a wrapper for the ExponentialGenerator for the org.uncommons.maths.random.ExponentialGenerator . It is used to produce pseudorandom numbers that follow the exponential distribution and can be used in the simulation for the frequency that the commands will be send to the robot and possibly for the frequency some sensors will send measurements. Its constructor, ExponentialFreqGenerator(int period, int repeat) has two arguments, the period being the timeframe during which we expect a number of commands or measurements with repeat being the number of commands or measurements. The time will then will be obtained by calling the getWaitTime() method which returns a long and should be passed as argument to the sleep method so that the input generators wait before they generate the next message.

4.4.4.2 Class: FileWifiReader

FileWifiReader class reads integers from a file named wifiquality.txt (each line has an integer) and stores them. When getWifiQuality() method is called it returns the next integer. If all the integers have been returned then 0 will be returned. This class implements the WifiQualityReader interface.

4.4.4.3 Interface: InputGenerator

4.4.4.4 Class: SensorDataGenerator

SensorDataGenerator class is used to generate random numbers and add them to a producer. It can generate data either with stable frequency or with a variable frequency that follows the exponential distribution. It has the following variables.

- boolean open: a flag to keep the SensorDataGenerator running untill it is set to false.
- OstProducer producer: the producer that will receive the data created by SensorDataGenerator.
- int period: the time period during which a repeat number of data will be produced
- int repeat: the number of data that will be produced every time period.
- Boolean exp: if set to true the data will be generated with a frequency that follows the exponential distribution, if false they will be produced over regular time periods.
- ExponentialFreqGenerator gen: the generator used to generate wait times that follow the exponential distribution.

4.4.4.5 Class: SimulationEnvironment

This class is responsible to simulate a real turtlebot session to test the optimal stopping algorithm. It instantiates a number of producer consumer pairs and then starts them. It was not required for the testing of the algorithm and the data generators haven't been utilized so it is incomplete.

4.4.5 Package: wifiQuality

4.4.5.1 Interface: WifiQualityReader

This interface is implemented from the classes that provide wifi quality data. In this thesis we use it to read the recorded data but can also be used to get live wifi quality data. Its only method getWifiQuality() returns an int with the quality of the wifi.

4.4.6 Recording wifi quality

During the development of the project we recorded the wifi quality directly from the linux os. The following code was used:

Table 4.2: SNR recording code

```
String command = "iwconfig | grep 'Link Quality'";
Process p = new ProcessBuilder(new String[] {"bash", "-c",
command}).start();
p.waitFor();
BufferedReader reader = new BufferedReader(new
InputStreamReader(p.getInputStream()));
String line=reader.readLine();
while(line != null){
     System.out.println(line);
```

```
     line=reader.readLine();
}
```

4.4.7 Multithreading

All the classes responsible for the production, the consumption of the messages and the decision making are implemented as seperate threads. This design choice allows us to support multiple sensors, each capturing data in a different frequency. Each sensor should have a different consumer sending its data to a different topic. We are able to prioritize some sensors above or below the others by creating and utilizing a different version of the decision maker For example if we have 3 devices with 50 sensors each each device will run 50 threads for the producers and 50 threads for the decision makers resulting in a total of 300 threads across those 3 devices.

4.5 Experimental Results/Simulations

For the experiments of this thesis we simulated the experimental setup using a laptop with intel core i3-3110M (2,4 Ghz, 3MB L3 cache) cpu and 4 GB DDR3 memory. The os used was Ubuntu 16.04. The IDE used for the code development was eclipse Neon.3 Release (4.6.3).

In this section, we report an experimental evaluation to compare the performance of the proposed framework. The experimental setup has as follows: we have used a UGV named ‘tutlebot’ and the role of GCS was handled by a fixed server. Turtlebot and GCS are part of the *Road-, Air- and Water-based Future Internet Experimentation* (RAWFIE) platform which offers a framework for interconnecting numerous testbeds over which remote experimentation can be realized.

Table 4.3: Simulation Parameters

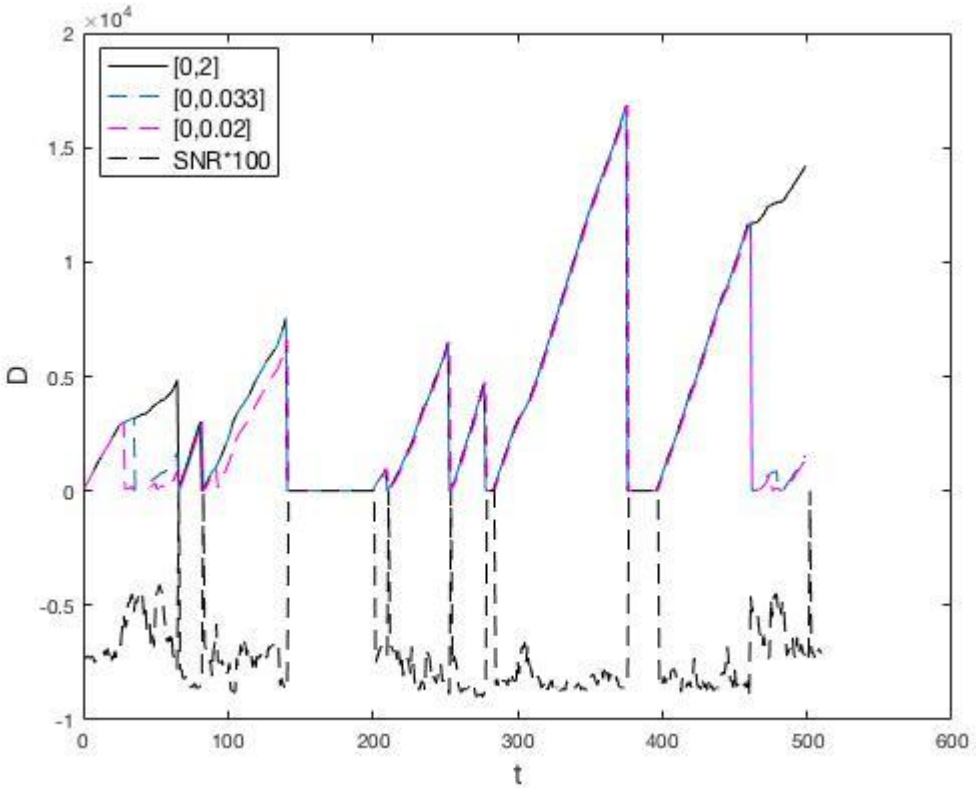
Value α	False Alarms	Total Alarms
[0,1]	4	8
[0,0.2]	3	8
[0,0.033]	1	8
[0,0.02]	2	11
[0,0.01]	5	11

RAWFIE platform originates in a European Union-funded (H2020 call: FIRE+ initiative) project which focuses on the mobille IoT paradigm and provides research and experimentation facilities through the ever growing domain of unmanned networked devices (vehicles).

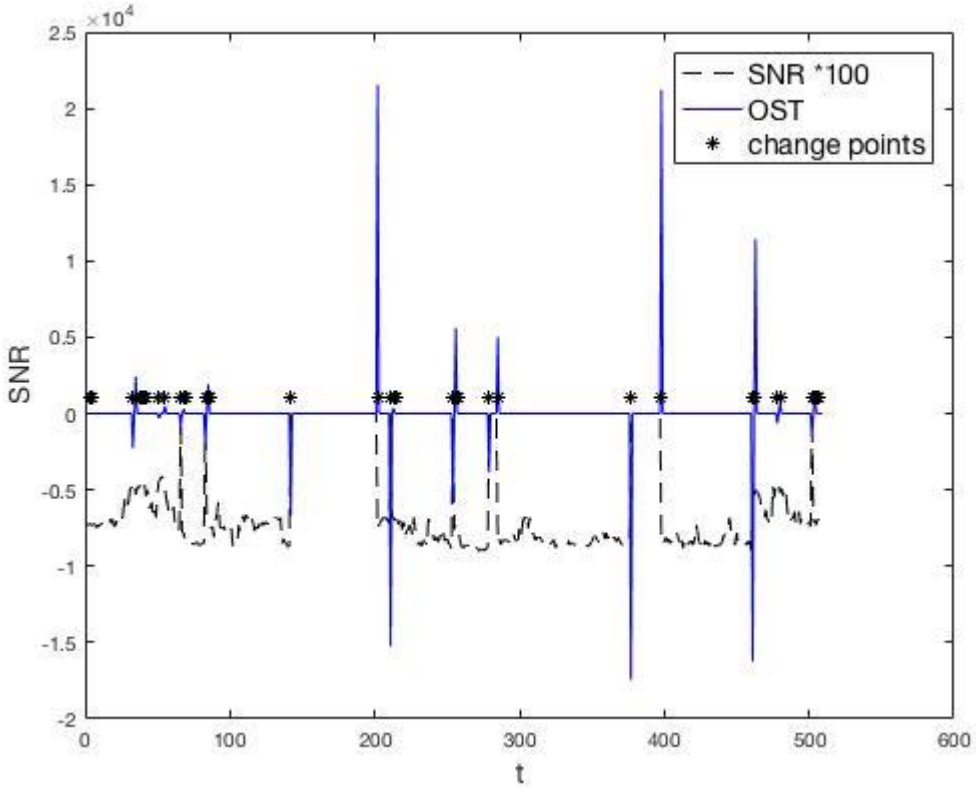
The mission contains a sequence of way-points in an outdoor infrastrure. In this infrastructure, as shown in Drawing 6, the turtlebot faces saturated network conditions.

GCS initializes the mission and sends commands to the UGV. Commands are being sent following the exponential distribution. When UGV approaches one goal point then the next “Goto” command is sent to the vehicle. UGV sends with a specific frequency $\mathcal{F}$messages. The profiling between a “good” and “bad” network quality distribution was pre-created as follows: For our first experiment we recorded the SNR once per second for a total duration of 2 hours, in which a mobile device is close to an AP with high bandwidth and gather SNR measurements. These measurements correspond to “good quality”. We measured for the same duration and frequency the SNR in the boundaries of AP in which the signal was weak and at some times completely lost or considerably saturated due to the over usage of bandwidth from other connected users, i.e., these figures characterized the quality of the network as “bad”.

The final output of the two distributions was shown in Drawings 8 and 9. The false alarm rate (FAR) in (8) was studied accordingly. We run a number of missions in order to study the performance of the OST decision module for the different values of α. Different values of α were evaluated by the number of the false alarms detected. The different α values were generated randomly in the intervals presented in Table I. The performance of the proposed OST decision making component was compared based on the different selected α values. We run the missions for N = 100 and measured the function D of equation (10). In Drawing 11, we can observe the changes of the mechanism based on the changes of SNR. The mean values of all the experiments were compared in the Drawing 11. In addition we multiplied the initial SNR values by 100 in order to be readable in the same figure. We can observe that the closest solution to the changes is the decision in the interval [0,0.02], in which the overall changes in the distributions are depicted also in the related plot.



Drawing 12: The value of the D function based on different values of η.



Drawing 13: OST decision making mechanism based on change detection between 'bad' and 'good' SNR.

Taking α values as granted, we run the missions of the UGV and can see the events generated by the optimal stopping decision making mechanism. We observe that in those locations with a small change in the distribution, it creates an alert for a change to the strategy and secures the safe execution of the mission. The messages that were paused, they were delayed in a queue and forwarded when the quality of the network was sufficient enough.

4.6 Related Work

In the literature researchers have extensively studied message-routing protocol employed on the unmanned vehicles. Opportunistic networks are studied as long as they are cappable of maintaining efficient operation in a wide range of network density and mobility conditions. In classifying the diversity of topological conditions in networking environments, one end of the spectrum corresponds to almost static dense topologies. For these, conventional topology-based protocols [11] function best; they use just node labels/identities. As the nodal density decreases and/or the mobility status between pairs of nodes remains stable, position-based families of protocols [12], [13] become more suitable. Protocols of this kind are based (in common with topology-based protocols) on the ‘forward’ action, i.e., the spatial transposition of messages due to hops from one node to another. They employ information about the nodes position/location (possibly in addition to identities/labels). Additionally in networks of low nodal density, intense mobility becomes a prerequisite for the creation of contact oportunities. For such topologies, protocols based on the ‘carry’ action [14], [15] , i.e., the spatial transposition of the message due to the physical movement of the carrier node, perform better. They employ information about the nodes motion characteristics. The aforementioned routing protocols have been designed to accommodate a restricted set of possible network conditions, corresponding to a particular sub-range and yield satisfactory performance only under these conditions. Clearly, improved frameworks, capable of coping with the full diversity of conditions, should resort to an appropriate, dynamically adjusted combination of both

the forward and carry actions. Another approach for resolving the connectivity issues in mobile devices and secure the delivery of the messages is to create a Desicion Making Process (DMP) by explicitly considering the heterogeneous multimedia traffic characteristics, e.g., delay deadlines, distortion impacts and dependencies etc. Researchers in [1] develop a cross-layer optimization framework for single-user multimedia trasmission over single hop wireless networks by considering network conditions and adaptation capability of the user at various layers of the protocol stack.

Going a step further we can study methods derived from the optimal stopping theory that have been applied to information dissemination in ad-hoc networks. However data delivery mechanisms have been studied in the literature from a different 'perspective' in mobile ad-hoc networks. The contexual data delivery mechanisms in [16]–[18] deal with the delivery of quality information to context-aware applications in static and mobile ad-hoc networks respectively assuming epidemic-based information dissemination schemes. The mechanism in [16] is based on the probabilistic nature of the "secretary problem" [5] and the optimal on-line problem. Authors in [17] study a dynamic video encoder that detects scene changes and tunes the synthesis of Groups-of-Pictures accordingly. Such dynamic encoding can be applied to infrastructures with restricted resources, like IoT facilities where multimedia streams are of use. They propose a time-optimized DMP that yields different sizes of groups-of-pictures (frames) to meet the previously discussed objectives i.e., transmit video sequences in acceptable quality with rational use of the wireless resources. In [19] authors propose optimal DMP decisions on the collection of contextual data from WSNs. The authors determine the best time to switch from decision to learning phase of Principal Component-based Context Compression (PC3) model, while data inaccuracy is taken into account. If data inaccuracy remains at low levels, then any deterministic switching from compression to learning phases of the observation.

Change-point detection has its origins almost sixty years ago in the work of Page [6] , Shiryayev [7] , and Lorden [8] , who focused on sequential detection of a change-point in an obseved stochastic process. The stochastic process was typically a model for the measured quality of a continuous production process, and the change-point indicated a deterioration in quality that must be detected and corrected. In our case, we are adopting this methodology for finding the best time instance for the controller strategy in order to maximize the possibility of successful message delivery by observing runtime network statistics.

5. Conclusion

This thesis describes a framework that monitors network conditions like signal strength and prioritizes the UxV transmission of control message and telemetry trying not to overload the network in which the devices operate and move. We propose a model of dynamic decision making, adaptive to changes in network conditions, by dynamically adjusting the transmission of control messages and telemetry based on an optimal stopping rule. The performance evaluation showed the automatic generation of alerts based on the change detection approach. At our future agenda we plan to extend our mechanism to a time-constraint approach with an additional OST problem of finite horizon in order to include a trade-off for the 'pausing' strategy. We can investigate how the GCS will trade-off the time pausing a command with the message delivery rate.

6. ABBREVIATIONS - ACRONYMS

AP	Access Point
CUMSUM	Cumulative Sum
DMP	Decision Making Process
FAR	False Alarm Rate
GCS	Ground Control Station
GoP	Group of Pictures
IMU	Inertial Measurement Unit
IoT	Internet of Things

OST	Optimal Stopping Theory
SNR	Signal to Noise Ratio
UGV	Unmanned Ground Vehicle
UxV	Unmanned Vehicle, x can stand for aerial, ground or sea
WSN	Wireless Sensor Network

7. REFERENCES

[1] F. Fu and M. van der Schaar, “Dependent optimal stopping framework for wireless multimedia transmission,” *2010 IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 2354–2357, 2010.

[2] A. Martinez and E. Fernández, *Learning ROS for Robotics Programming*. Packt Publishing, 2013.

[3] J. M. O’Kane and J. M. O. Kane, *A gentle introduction to ROS*. 2013.

[4] “Sensors supported by ROS.” [Online]. Available: http://wiki.ros.org/Sensors#Sensors_supported_by_ROS. [Accessed: 09-Jan-2018].

[5] T. Ferguson, *Optimal stopping and applications*. Mathematics Department, UCLA.

[6] E. S. Page, “Continuous Inspection Schemes,” *Biometrika*, vol. 41, no. 1/2, p. 100, 1954.

[7] A. bert N. Shirayaev and A. B. Aries, *Optimal stopping rules*, no. 8. 2008.

[8] G. Lorden, “Procedures for Reacting to a Change in Distribution,” *Ann. Math. Stat.*, vol. 42, no. 6, pp. 1897–1908, 1971.

[9] J. Unnikrishnan, V. V. Veeravalli, and S. Meyn, “Least favorable distributions for robust quickest change detection,” *IEEE Int. Symp. Inf. Theory - Proc.*, pp. 649–653, 2009.

[10] G. V. Moustakides, “Optimal Stopping Times for Detecting Changes in Distributions,” *Ann. Stat.*, vol. 14, no. 4, pp. 1379–1387, 1986.

[11] A. B. Mcdonald, “Survey of Adaptive Shortest-Path Routing in Dynamic Packet-Switched Networks 1 Introduction,” pp. 1–29, 1997.

[12] Min Chen, V. C. M. Leung, Shiwen Mao, Yang Xiao, and I. Chlamtac, “Hybrid Geographic Routing for Flexible Energy—Delay Tradeoff,” *IEEE Trans. Veh. Technol.*, vol. 58, no. 9, pp. 4976–4988, Nov. 2009.

[13] S. Giordano, I. Stojmenovic, and L. Blazevie, “Position Based Routing Algorithms for Ad Hoc Networks: a Taxonomy,” *Ad Hoc Wirel. Netw.*, pp. 103–136, 2003.

[14] Y. Cao and Z. Sun, “Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges,” *IEEE Commun. Surv. Tutorials*, vol. 15, no. 2, pp. 654–677, 2013.

[15] H. M. Lin, G. Yu, A. C. Pang, and J. S. Pathmasuntharam, “Performance study on delay tolerant networks in maritime communication environments,” *Ocean. IEEE Sydney, Ocean. 2010*, 2010.

[16] C. Anagnostopoulos and S. Hadjiefthymiades, “Delay-tolerant delivery of quality information in ad hoc networks,” *J. Parallel Distrib. Comput.*, vol. 71, no. 7, pp. 974–987, 2011.

[17] K. Panagidi, C. Anagnostopoulos, and S. Hadjiefthymiades, “Optimal grouping-of-pictures in iot video streams.”

[18] C. Anagnostopoulos and S. Hadjiefthymiades, “Optimal, quality-aware scheduling of data consumption in mobile ad hoc networks,” *J. Parallel Distrib. Comput.*, vol. 72, no. 10, pp. 1269–1279, 2012.

[19] C. Anagnostopoulos and S. Hadjiefthymiades, “Advanced Principal Component-Based Compression Schemes for Wireless Sensor Networks,” *Acm Trans. Sens. Networks*, vol. 11, no. 1, pp. 1–34, 2014.