NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

PAULO CÉSAR LEITE DE MATOS

BSc in Computer Science and Engineering

FAULT-TOLERANCE IN A PUBLISH/SUBSCRIBE SYSTEM WITH MULTIPLE DELIVERY GUARANTEES

SOME THOUGHTS ON THE LIFE, THE UNIVERSE, AND EVERYTHING ELSE

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon

Draft: *November 28, 2023*

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

FAULT-TOLERANCE IN A PUBLISH/SUBSCRIBE SYSTEM WITH MULTIPLE DELIVERY GUARANTEES

SOME THOUGHTS ON THE LIFE, THE UNIVERSE, AND EVERYTHING ELSE

PAULO CÉSAR LEITE DE MATOS

BSc in Computer Science and Engineering

Adviser: Hervé Miguel Cordeiro Paulino
Associate Professor, NOVA School of Science and Technology

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
Draft: *November 28, 2023*

Abstract

Regardless of the language in which the dissertation is written, usually there are at least two abstracts: one abstract in the same language as the main text, and another abstract in some other language.

The abstracts' order varies with the school. If your school has specific regulations concerning the abstracts' order, the NOVAthesis LaTeX (novathesis) (LaTeX) template will respect them. Otherwise, the default rule in the novathesis template is to have in first place the abstract in *the same language as main text*, and then the abstract in *the other language*. For example, if the dissertation is written in Portuguese, the abstracts' order will be first Portuguese and then English, followed by the main text in Portuguese. If the dissertation is written in English, the abstracts' order will be first English and then Portuguese, followed by the main text in English. However, this order can be customized by adding one of the following to the file `5_packages.tex`.

```
\ntsetup{abstractorder={<LANG_1>,...,<LANG_N>}}
\ntsetup{abstractorder={<MAIN_LANG>={<LANG_1>,...,<LANG_N>}}}
```

For example, for a main document written in German with abstracts written in German, English and Italian (by this order) use:

```
\ntsetup{abstractorder={de={de,en,it}}}
```

Concerning its contents, the abstracts should not exceed one page and may answer the following questions (it is essential to adapt to the usual practices of your scientific area):

1. What is the problem?
2. Why is this problem interesting/challenging?
3. What is the proposed approach/solution/contribution?
4. What results (implications/consequences) from the solution?

Keywords: One keyword, Another keyword, Yet another keyword, One keyword more, The last keyword

Resumo

Independentemente da língua em que a dissertação está escrita, geralmente esta contém pelo menos dois resumos: um resumo na mesma língua do texto principal e outro resumo numa outra língua.

A ordem dos resumos varia de acordo com a escola. Se a sua escola tiver regulamentos específicos sobre a ordem dos resumos, o template (LaTeX) novathesis irá respeitá-los. Caso contrário, a regra padrão no template novathesis é ter em primeiro lugar o resumo *no mesmo idioma do texto principal* e depois o resumo *no outro idioma*. Por exemplo, se a dissertação for escrita em português, a ordem dos resumos será primeiro o português e depois o inglês, seguido do texto principal em português. Se a dissertação for escrita em inglês, a ordem dos resumos será primeiro em inglês e depois em português, seguida do texto principal em inglês. No entanto, esse pedido pode ser personalizado adicionando um dos seguintes ao arquivo `5_packages.tex`.

```
\abstractorder(<MAIN_LANG>):={<LANG_1>,...,<LANG_N>}
```

Por exemplo, para um documento escrito em Alemão com resumos em Alemão, Inglês e Italiano (por esta ordem), pode usar-se:

```
\ntsetup{abstractorder={de={de,en,it}}}
```

Relativamente ao seu conteúdo, os resumos não devem ultrapassar uma página e frequentemente tentam responder às seguintes questões (é imprescindível a adaptação às práticas habituais da sua área científica):

1. Qual é o problema?
2. Porque é que é um problema interessante/desafiante?
3. Qual é a proposta de abordagem/solução?
4. Quais são as consequências/resultados da solução proposta?

Palavras-chave: Primeira palavra-chave, Outra palavra-chave, Mais uma palavra-chave, A última palavra-chave

Contents

List of Figures v

List of Tables vi

Glossary vii

Acronyms viii

Symbols ix

1 Introduction 1

1.1 Welcome to the novathesis Template . . . 1

1.1.1 Your Time is Precious . . . 1

1.1.2 Recognition . . . 2

1.2 The *NOVAthesis* Template . . . 2

1.3 Getting Started . . . 7

1.3.1 Using Overleaf . . . 7

1.3.2 Using a Local LaTeX Installation Local . . . 8

1.4 Getting Help . . . 8

1.4.1 Suggestions, Bugs and Feature Requests . . . 9

1.5 Donors . . . 9

1.6 Disclaimer . . . 9

2 Background 10

2.1 Replication . . . 10

2.1.1 Total and Partial Replication . . . 10

2.2 Consistency . . . 11

2.2.1 Consistency Models . . . 11

2.2.2 Data-centric consistency models . . . 12

2.2.3 Client-centric consistency models . . . 13

2.3 Publish/Subscribe Systems . . . 14

2.3.1 Event Service . . . 14
2.3.2 Subscription schemes . . . 15
2.3.3 Reliable Publish/Subscribe Systems . . . 16
2.4 Example glossary, acronyms, and symbols . . . 20

3 A Short LaTeX Tutorial with Examples **23**
3.1 Document Structure . . . 23
3.2 Dealing with Bibliography . . . 23
3.3 Inserting Tables . . . 23
3.4 Importing Images . . . 23
3.5 Floats, Figures and Captions . . . 23
3.6 Text Formatting . . . 25
3.7 Generating PDFs from LaTeX . . . 25
3.7.1 Generating PDFs with pdflatex . . . 25
3.7.2 Dealing with Images . . . 26
3.7.3 Creating Source Files Compatible with both latex and pdflatex . 26
3.8 Equações . . . 28
3.9 Test for algorithms . . . 28

Bibliography **32**

Appendices

A *NOVAthesis* covers showcase **35**

B Appendix 2 Lorem Ipsum **36**

Annexes

I Annex 1 Lorem Ipsum **38**

List of Figures

1.1 The novathesis project web page in GitHub. 4
1.2 NOVAthesis template in Overleaf. 7
1.3 The NOVAthesis Project page in GitHub. 8

2.1 Publish/Subscribe architecture 15

3.1 A figure with two sub-figures! 24
3.2 Imagem em formato *bitmap* (JPG) 29
3.3 Imagem em formato PDF vectorial 30
3.4 Exemplo de utilização de *subbottom* 31

List of Tables

1.1 NOVA University Lisbon's Schools supported by the novathesis template . . 2
1.2 University of Lisbon's Schools supported by the novathesis template 4
1.3 University of Minho's Schools supported by the novathesis template 4
1.4 Instituto Politécnico de Lisboa's Schools supported by the novathesis template 6
1.5 Instituto Politécnico de Setúbal's Schools supported by the novathesis template 6
1.6 Other Universities/Schools/Degrees's Schools supported by the novathesis template 7

2.1 Fault models based on [13]. 16
2.2 Solution properties. 21

3.1 Test results summary. 27

Glossary

computer An electronic device which is capable of receiving information (data) in a particular form and of performing a sequence of operations in accordance with a predetermined but variable set of procedural instructions (program) to produce a result in the form of information or signals. This is a test that adds a citation [**Artho04**] to the glossary! *(p. 22)*

Acronyms

aaa	acornym aaa *(p. 20)*
aab	acornym aab *(p. 20)*
aba	acornym aba *(p. 20)*
abbrev	abbreviation of a longer text *(p. 20)*
bbb	acornym bbb *(p. 20)*
DI	Department of Computer Science *(p. 2)*
FCT	NOVA School of Science and Technology *(pp. 2, 10)*
NOVA	NOVA University Lisbon *(p. 2)*
novathesis	NOVAthesis LaTeX *(pp. i, ii, v, vi, 1, 2, 4, 6–10)*
pubsub	publish/subscribe *(pp. 14–18, 20)*
xpto	and extension of a xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto *(p. 20)*

Symbols

μ Mu *(p. 22)*

π the numerical value of pi *(p. 20)*

r the radius of a circle *(p. 20)*

1 Introduction

This is the novathesis LaTeX template Version 7.1.5 from Template!date2023-08-14.

1.1 Welcome to the novathesis Template

This first Chapter introduces the novathesis template and how it is organized. In Chapter you can find some specific instructions on how to use this template. Chapter 3 shows some examples and give some hints on how to write your text. Please read these next Chapters carefully.

1.1.1 Your Time is Precious

Did you learn how to drive by sitting by the wheel and throwing your car into the road? Most probably you did take your time *learning the rules* and *practicing* first! Likewise, it is not wise to throw yourself at the task of writing a thesis/dissertation in LaTeX without seriously considering the following recommendation!

> If you are going to spend zillions of hours writing your thesis/dissertation using the novathesis LaTeX template (or some other LaTeX template), be wise and spend a couple of hours learning how to use it properly by reading its manual. And then, be even wiser, and spend a few more hours learning some LaTeX. I am sure that the time you are investing now will pay itself countless times before you submit your thesis/dissertation.
>
> — *João Lourenço*

1.1.2 Recognition

The novathesis tempalte was born in 1996, and what you see now accumulates to many many hundreds (thousands?!) of working hours, unpaid and stollen from family and friends. This work is available to the community under the LaTeX Project Public License v1.3c, which means you are entitled to use it for free and change it at your will. However, if you decide to use this template to write your thesis/dissertation, **be fair to the developers** and:

1. Cite the novathesis manual [16] in a place of your choice (e.g., in the *Acknowledgments*) of your thesis/disseration with "`\cite{novathesis-manual}`" . If you cite it this way, the correct entry will be added automatically to your bibliography (no need to worry with the necessary BibTeX entry, as it will be added automatically);

2. Go to the project web page in GitHub and give the project a star (marked with a red ellipse at the top-right in Figure 1.1); and

3. Make a donation by visiting the novathesis project page and clicking in the button marked with a green ellipse at the top-center in Figure 1.1). Alternatively, just click **HERE** and your browser will be directed to the right page.

1.2 The *NOVAthesis* Template

The novathesis template was born at the Department of Computer Science (DI) of NOVA School of Science and Technology (FCT) of NOVA University Lisbon (NOVA), Portugal. But the user base grew... initially grew to other Departments of FCT-NOVA, then to other Schools of NOVA, and later to other Schools of other Universities. Currently more than 25 Schools are natively supported by the novathesis template (see Tables 1.1, 1.2, 1.3, 1.4, 1.5, and 1.6).

Table 1.1: NOVA University Lisbon's Schools supported by the novathesis template

NOVA University Lisbon

NOVA School of Science and Technology (FCT-NOVA)

- All PhD Programs (PhD)
- All MSc Programs (MSc)

NOVA School of Social Sciences and Humanities (FCSH-NOVA)

- All PhD Programs (PhD)
- All MSc Programs (MSc)

NOVA Information Management School (NOVA-IMS)

- All PhD Programs (PhD)
- Master's in Data Science and Advanced Analytics (MMAA)
- Master's in in Statistics and Information Management (MEGI)
- Master's in Information Management (MGI)
- Master's in Geographic Information Systems and Science (MCSIG)
- Master's in Geospatial Technologies (GeoTech)

National School of Public Heath (ENSP-NOVA)

- All PhD Programs (PhD)
- All MSc Programs (MSc)

Instituto de Tecnologia Química e Biológica (ITQB-NOVA)

- All PhD Programs (PhD)
- All MSc Programs (MSc)

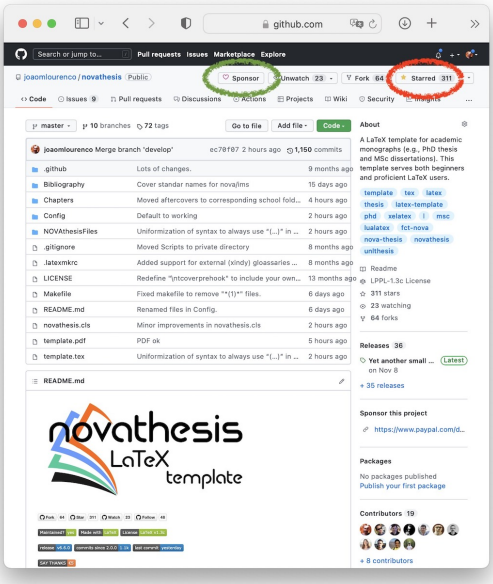


Figure 1.1: The novathesis project web page in GitHub.

Table 1.2: University of Lisbon's Schools supported by the novathesis template

University of Lisbon	
	**Instituto Superior Técnico** (IST-UL) • All PhD Programs (PhD) • All MSc Programs (MSc)
	**Faculdade de Ciências** (FCUL) • All PhD Programs (PhD) • All MSc Programs (MSc)
	**Faculdade de Medicina Veterinária** (FMV-UL) • All PhD Programs (PhD) • All MSc Programs (MSc)

Table 1.3: University of Minho's Schools supported by the novathesis template

University of Minho

School of Architecture (EA-UMINHO)
- All PhD Programs (PhD)
- All MSc Programs (MSc)

School of Sciences (EC-UMINHO)
- All PhD Programs (PhD)
- All MSc Programs (MSc)

School of Law (ED-UMINHO)
- All PhD Programs (PhD)
- All MSc Programs (MSc)

School of Engineering (EE-UMINHO)
- All PhD Programs (PhD)
- All MSc Programs (MSc)

School of Economics and Management (EEG-UMINHO)
- All PhD Programs (PhD)
- All MSc Programs (MSc)

School of Medicine (EM-UMINHO)
- All PhD Programs (PhD)
- All MSc Programs (MSc)

School of Psychology (EP-UMINHO)
- All PhD Programs (PhD)
- All MSc Programs (MSc)

School of Nursing (ESE-UMINHO)
- All PhD Programs (PhD)
- All MSc Programs (MSc)

	**Institute of Social Sciences** (ICS-UMINHO) • All PhD Programs (PhD) • All MSc Programs (MSc)
	**Institute of Education** (IE-UMINHO) • All PhD Programs (PhD) • All MSc Programs (MSc)
	**School of Arts and Humanities** (ELACH-UMINHO) • All PhD Programs (PhD) • All MSc Programs (MSc)
	**Research Institute 13Bs** (I3B-UMINHO) • All PhD Programs (PhD) • All MSc Programs (MSc)

Table 1.4: Instituto Politécnico de Lisboa's Schools supported by the novathesis template

Instituto Politécnico de Lisboa

	**Instituto Superior de Engenharia de Lisboa** (ISEL-IPL) • All MSc Programs (MSc)

Table 1.5: Instituto Politécnico de Setúbal's Schools supported by the novathesis template

Instituto Politécnico de Setúbal

	**Escola Superior de Tecnologia de Setúbal** (ISEL-IPL) • All MSc Programs (MSc)

Table 1.6: Other Universities/Schools/Degrees's Schools supported by the novathesis template

Other Universities/Schools/Degrees
 **Escola Superior de Enfermagem do Porto** (ESEP) • All MSc Programs (MSc)

1.3 Getting Started

The template provides an *easy to use* setting for you to write your thesis/dissertation in LaTeX:

- Select your school;
- Fill your thesis metadata (title, research field, etc) in the file "`template.tex`";
- Create your thesis/dissertation contents using the files in folder "`Chapters`"; and
- Process using you favorite LaTeX processor (pdfLaTeX, XeLaTeX or LuaLaTeX).

1.3.1 Using Overleaf

Overleaf is a collaborative cloud-based LaTeX editor used for writing, editing and publishing scientific documents. Like "Goggle Docs", for LaTeX users"). You can edit and compile your LaTeX source on the cloud, without installing software in your own computer, and, much like *Google Docs*, you can share your document with others users and everybody can edit the same file at the same time (this may be dangerous).

If you do not have an account in Overleaf, you must create one first.

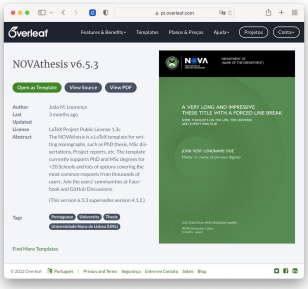


Figure 1.2: NOVAthesis template in Overleaf.

Once you have an account, please access the novathesis template in Overleaf and select the green button *Open as Template* (see Figure 1.2).

Please notice that the version currently available in Overleaf (v6.10.10) is slightly outdated (current version is v7.1.5). A new version (v7.0.0) will be submitted to Overleaf soon. Until then, please:

1. *Download the latest version from the GitHub repository as a Zip file.*

2. *Login to your favorite LaTeX cloud service. I recommend Overleaf but there are alternatives (these instructions apply to Overleaf and you'll have to adapt for other providers).*

3. *In the menu select:* `New project` → `Upload project`.

4. *Upload the zip file.*

5. *Select "template.tex" as the main file.*

6. *Let Overleaf compile the document.*

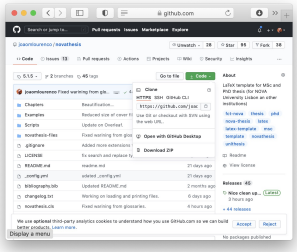


Figure 1.3: The NOVAthesis Project page in GitHub.

1.3.2 Using a Local LaTeX Installation Local

First of all, start by installing LaTeX in your computer. There are two main distributions, MikTeX and TeXLive, and both of them are available for the 3 most popular Operating Systems: Linux, macOS and Windows.

Be aware that a full installation of MikTeX or TeXLive will take near 5 GB of hard disk space. So, think twice before installing the full distribution. See the novathesis Wiki for the list of packages required to compile the template.

Once you have LaTeX up and running, remember to install a good LaTeX text editor. I recommend you to take a look at this post in the `tex.stackexchange.com` site. If you want a quick and dirty recommendation, try TeXStudio.

Now, you must access the novathesis repository in GitHub, select the green button *Code* and then *download* (or *clone*) the template. You will always get the latest version of the template (currently v7.1.5 from 2023-08-14).

1.4 Getting Help

No! You don't have to use this template to write your thesis. You don't even have to use LaTeX. However, writing a thesis is serious stuff, and which tool you shall use to write it is not a decision to make lighthearted.

LaTeX is hard enough by itself. This template aims at making your life easier, but not easy. If you choose to use this template to write your thesis, you are very welcome. However, don't expect me to provide you help with LaTeX. Look for help with your friends (you have some friends, don't you?), or search the web, or try even to read some book(s) on LaTeX. In the end you will certainly find the experience rewarding.

When you come to the point of "*How do I do this with the* novathesis *template?*", remember...

1. To check the novathesis wiki and have some hope! `:D`

2. Google is your best friend.

3. Search the GitHub Discussions page for a question related to yours. *If and only if* you don't find one, then post your own question in English please!

4. Search the NOVAtheis Facebook group for a question related to yours. *If and only if* you don't find one, then post your own question in either Portuguese or English, at your preference.

When you post your own question, remember to **always** state the novathesis version number you are using and referring to.

Please do not attempt to contact me directly (email, Messenger, etc)...
I WILL NOT REPLY!

1.4.1 Suggestions, Bugs and Feature Requests

Help: If you just need some help, see above Section 1.4.

Suggestion: Do you have a suggestion/recommendation? Please add it to the wiki and help other users!

Bug: Did you find a bug? Please open an issue. Thanks!

New Feature: Would you like to request a new feature (or support of a new School)? Please open an issue. Thanks!

1.5 Donors

The list of *Donnors* is available in the novathesis Project page.

1.6 Disclaimer

Although the novathesis template is endorsed by some Schools (e.g., linked from FCT-NOVA web site), the novathesis template **this not an official template** for any School.

The novathesis template exists to make your life easier and we do our best to make it compliant to the supported (+25) Schools' regulations but, in the end of the line, you and only you are accountable for both the look and the contents of the document you submit as your thesis/dissertation.

2

Background

This Chapter describes how to use the novathesis template. It is assumed that you have a working of LaTeX, either local (in your own computer) or remote (in Overleaf), and that you were able to generate a PDF for the default configuration of the template: a PhD thesis for FCT. This Chapter aims at exemplifying how to do common stuff with LaTeX. We also show some stuff which is not that common! ;)

Please, use these examples as a starting point, but you should always consider using the *Big Oracle* (aka, Google, your best friend) to search for additional information or al-ternative ways for achieving similar results.

2.1 Replication

Replication is one of the cornerstones in the design of reliable distributed systems. It is a widely used technique to provide high availability, low latency and the ability to tolerate individual node losses (fault-tolerance). A replicated system is one in which multiple devices work together to replicate the behavior of the same system as if only a single device was responsible for the entirety of it.

There are several replication models and techniques one can choose when designing a replicated database system. Each approach to the system design can provide different properties to it.

2.1.1 Total and Partial Replication

When thinking about data location and availability, one can reason about the total and partial replication models.

With total replication, every instance of the database has a complete copy of it's state, meaning that every storage node contains all data available in the system. With this approach one can achieve high data availability, since data is available in every database instance, faster query execution and improved load balancing, since queries can be executed in any replica, especially when the system's clients are scattered throughout the world, by having multiple instances of the data closer to the clients. On the other hand,

since total replication requires that all replicas have their data synchronized at all times, it might introduce the problem of data consistency and slower update process.

In contrast, in the partial replication model, each replica only holds a subset of the database [19]. This can be an interesting approach when storage space is a concern because updates are propagated only towards the affected sites. However, certain sites are not able to execute certain types of transactions, which can affect the ability to load balance.

2.1.1.1 Active and Passive Replication

The previous concepts, are mainly focused on addressing where data is being stored. However, it is also necessary to consider how data is replicated and how updates are propagated among the different replicas. The active and passive replication models provide two different methods of approaching this issue.

Active replication is a method in which client operations are forwarded to all replicas in a coordinated order, relying on a protocol to decide the order of execution of such operations. Afterwards, each replica executes the operations independently, reaching a result state identical among all replicas. With this approach, to guarantee that replicas state after executing all operations is identical, client operations are required to be deterministic [15].

When using passive replication, all client requests are sent to a primary node, which executes the requests and sends update messages, containing the state of the master after executing the request, to the secondary nodes in the same order as the order of the operations executed by the master replica. Unlike active replication, client operations are not required to be deterministic due to the fact that only the master node receives and executes requests. On the other hand, this approach can consume a lot of bandwidth when the result state is large. Also, the primary node might fail and, thus, this method is required to have a replica recovery and master election protocol [4].

2.2 Consistency

A concerning aspect of replication is data consistency. It is required that replica's state is consistent among all of them. Replica's state being consistent means that if a client executes an operation, the result state of all replicas is identical, independently of which replica receives the client's request. This can become an even harder challenge when multiple clients execute multiple operations in different replicas, especially if some of the operations are write operations. In that case, it is necessary to have a mechanism in place that assures that replicas always converge to the same state.

2.2.1 Consistency Models

In distributed systems, a consistency model refers to the guarantees provided by the system about the order in which operations appear to occur to clients. It determines how data is accessed and updated across multiple nodes, and how these updates are made

available to clients [1]. There are several consistency models, each with its advantages and disadvantages, and can be categorized in two categories: data-centric, client-centric [1].

When discussing consistency, read and write are the available operations to perform over a shared data item. When a process performs a read operation, it expects the return value to be the result of the last write operation performed on the given data item. However, it is not possible to have a global clock to synchronize operations and decide which write was performed last. Therefore, we need to provide alternatives that lead to a range of consistency models [20].

2.2.2 Data-centric consistency models

In the perspective of the data store, shared data is required to be stored consistently in all replicas. In order to achieve such requirement, it is necessary to establish protocols able to guarantee that data is up to date at all times in all replicas. Hence, data-centric models focus on the internal synchronization processes and the communication between replicas [3].

2.2.2.1 Strong Consistency Models

A system provides strong consistency if, at all times, data is consistent among all replicas. This means that after receiving and executing client requests, all replicas should converge to a single consistent state, by executing the same request before replying to the client. With this type of consistency model, clients can send their requests to any node because they will always get the exact same response to a request, unless faults occurred in the system or in nodes. The downside of this kind of consistency models is that the synchronization requirements might be too demanding, which can negatively impact response latency because clients only get a response after the synchronization process has terminated.

2.2.2.2 Linearizability

One of the strongest consistency models is ***Linearizability***, in which, each operation has an associated point in time, called linearization point, which is the moment in time when the operation appears to take effect atomically, without the interference of other concurrent operations. Under linearizability, it is guaranteed that the systems behaves as if all operations ocurred in a sequential manner [12].

2.2.2.3 Causal Consistency

In the causal consistency model, operations have a causal relationship between them, i.e. if an operation A is dependent on the output of an operation B, then operation B must be executed before operation A. For example, if operation B reads a resource and operation A updates the value of the resource based on the value read by B, then A must

be executed after B because A depends on B. With this model, it is guaranteed that a client does not read two related write operations in a wrong order and in case the client has read the latest value, then it does not read the stale value [1].

2.2.2.4 Weak Consistency

A weak consistency model is one that provides the lowest possible guarantees on the ordering of operations. In simpler terms, weak consistency can be seen as an approach where replicas might by chance become consistent [2].

2.2.3 Client-centric consistency models

Client-centric consistency insures that while a client has access to a resource, it will never see an inconsistent state of the resource. However, if multiple clients have access to the same resource simultaneously, then consistency is not guaranteed [1].

2.2.3.1 Eventual Consistency

Eventual consistency guarantees that when a write operation is performed over a data store, that operation's result will eventually be reflected in all replicas where the updated data item is stored(tese luis chula).

This model is a form of weak consistency, thus it isn't required to synchronize replicas when clients send requests allowing for reduced latency. After a period of time, if no faults occur in the system or no write operations where performed, then eventually all replicas will converge to a single consistent state. However, since updates are performed asynchronously, if other clients read the data item from other replicas, it is not guaranteed that the operation will return the most recent write.

2.2.3.2 Monotonic Reads

Monotonic reads ensure that when a client reads a version n of a data item, future read operations on the same replica will always return a version of the data item equal or newer than n [2]. From an application perspective data visibility might not be instantaneous but versions become visible in chronological order.

2.2.3.3 Monotonic Writes

It is important that write operations are propagated in the correct order to all copies of a data store. In a monotonic write consistent store, a write operation by a process on a data item k is completed before successive write operation on k by the same process. Therefore, a write operation on a copy of k is performed only if that copy has been brought up to date by means of any preceding write operation, which may have taken place on other copies of k. In case old writes have not finished, the new write must wait for old ones to finish [20].

2.2.3.4 Read Your Writes

A data store provides read-your-writes consistency, if the effect of a write operation by a process on data item *k* will always be seen by a successive read operation on *k* by the same process(distributed systems principles and paradigms). This condition guarantees that a write operation is always completed before a successive read operation by the same process, no matter where that read operation takes place.

2.2.3.5 Writes Follow Reads

Write follows reads consistency guarantees that any successive write operation by a process on a data item *k* will be performed on a copy of *k* that is up to date with the value most recently [20].

A data store provides write follows reads consistency, if a write operation by a process on a data item *k* following a previous read operation on *k* by the same process is guaranteed to take place on the same or a more recent value of *k* that was read

2.3 Publish/Subscribe Systems

A publish/subscribe (pubsub) system is one in which clients communicate asynchronously between them by exchanging notifications, relying on a notification service, also known as *broker* interposed between them [17]. In this systems there are two type of clients, producers and consumers, more often called publishers ans subscribers, respectively.

Producers publish events and consumers subscribe to events by issuing subscriptions. Consumers can have multiple active subscriptions and after issuing one, the event service is responsible for delivering all future notifications when publishers publish an event, subscribed by the subscriber, until the subscription gets canceled. This allows consumers to express their interest in an event or pattern of events. Figure 2.1 shows the architecture of pubsub systems with a single broker.

This kind of system is a great choice for applications focused on information dissemination, such as newsletters.

2.3.1 Event Service

The pubsub system model relies on an event notification service responsible for storing publications, managing subscriptions and efficiently delivering notifications to subscribers [11]. It acts as a middle-man between publishers and subscribers allowing these to be decoupled from each other.

The decoupling provided can be decomposed into three dimension [11]:

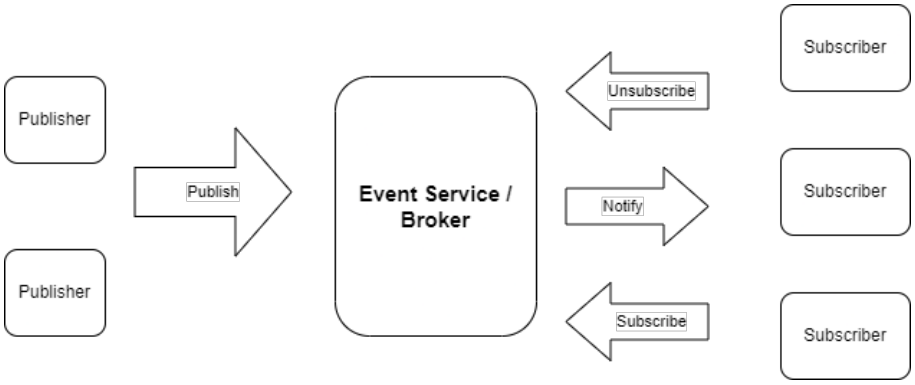


Figure 2.1: Publish/Subscribe architecture

- **Space decoupling**: publishers and subscribers are not required to know each other by relying on the event service to receive publications from publishers and delivering notifications to subscribers;
- **Time decoupling**: publishers and subscribers do not need to actively participate in the interaction at the same time. Publishers can publish events while subscribers are disconnected and subscribers can be notified of events while the original publisher of the event is disconnected;
- **Synchronization decoupling**: publishers do not get blocked while publishing events and subscribers get notified asynchronously of an event while performing other concurrent tasks.

2.3.2 Subscription schemes

Subscribers are often more interested in a specific set of events rather than all events. Thus, subscriptions had to be adapted to allow subscribers to specify the set of events they were most interested, which lead to the creation of several subscription schemes. The most widely used subscription schemes are the following [11]:

- **Topic-based subscription**: participants publish events and subscribe to individual topics, identified by keywords;
- **Content-based subscription**: consumers subscribe to events by specifying filters to event's content. Filters are rules that must be met by the content of a publication in order to get notified by the event service.
- **Type-based subscription**: extends topic-based subscription by introducing the notion of types.

2.3.2.1 Topic-Based Publish/Subscribe

This thesis will mainly focus on a topic-based pubsub system, therefore, this section goes into more detail about this type of system.

In topic-based systems, messages are associated with topics and are routed selectively to destinations with matching interests. Topics are associated with channels, which are queues of messages maintained by brokers, and it's name is used to create the channel. After creation, every channel is uniquely identified by it's name, also used as an argument for the publish() and subscribe() functions. Clients can subscribe to multiple topics and will receive future messages published to those topics [21].

As stated in [11], the concept of topics is similar to the concept of groups in the context of group communication. Also, subscriptions can be seen as joining a group and publishing a message as broadcasting that message to the members of that group.

2.3.3 Reliable Publish/Subscribe Systems

When dealing with distributed systems, it is generally required to add components to systems due to higher service demands and geographical positioning of clients. However, the more components a system is composed of, the more complex it becomes and the probability of components failing also increases.

In general, in distributed systems, **faults** can be defined as the deviation of a component from it's correct behavior. In order to develop a reliable system, it is important to make assumptions about how a process might fail. Therefore, it is of extreme importance to define a fault model. Table 2.1 presents some of the most commom fault models considered in existing systems.

Table 2.1: Fault models based on [13].

Type	Description
Crash Model	When a process fails it stops sending messages
Omission Model	After failing a process omits the trasmission or reception of any number of messages
Fail-Stop Model	Upon failure, the process "notifies" all other processes of its own failure and then stops sending messages
Byzantine Model	A process might deviate from its behavior in an arbitrary way

Detecting failures can seem to be a trivial task, however when dealing with very wide systems, it might present itself as a bigger challenge. Therefore, it is important to design robust systems and algorithms able to guarantee the correct behavior of correctly operating processes, in the presence of such failures.

In the context of distributed pubsub systems, it is often necessary to provide strong guarantees about the reliable delivery of information. This is mainly due to publishers and subscribers being decoupled as mentioned in section 2.3. Systems can rely, for example, on overlay networks of distributed brokers or group communication and reliable application-layer multicast, such as Scribe [5], to propagate messages between brokers and from publishers to subscribers [11].

Moreover, we should also take into account that brokers might eventually fail and, therefore, system's need to implement mechanisms to detect such failures and recover from them.

In [10], the authors highlight some techniques and strategies that have been adopted by current implementations:

- **Planning**: estimate the reliability level of several paths from a source to a destination, and pick the one that exposes the highest reliability. This mainly aims at avoiding faults as much as possile;
- **Reconfiguration**: adopt topological reconfiguration to recover connectivity in the forwarding tree after a node or link crashes. It aims at maintaining a consistent connectivity for the system;
- **Retrasmissions**: publishers store produced events so that subscribers can request retransmissions when losses are somehow detected;
- **Epidemic Algorithms**: each process exchanges at a random time its history of the received notifications with a randomly-chosen set of processes among the ones constituting the system. Subsequently, inconsistencies, i.e., message drops are detected by comparison and corrected through retransmissions;
- **Forward Error Correction**: forward redudant data instead of using retransmissions. This way the receiver can reconstruct te original event even if some notifications have been dropped during delivery;
- **Broker Replication**: replicates the brokers in the notification service. The state of a broker is replicated to its neighbors so in case of a broker failure, it can be easily substituted without losing subscription consistency;
- **Path Redundancy**: establishes redundant pathes among the nodes of the system. Notifications are sent through multiple paths, and only the first-received replica of the message is delivered to the

The following sections go over some approaches taken by researchers to address the reliability problem of pubsub systems.

2.3.3.1 Epidemic Algorithms for Reliable Content-Based Publish-Subscribe: An Evaluation

In the papers [8] and [9], the authors approach reliability by relying in gossip (or epidemic) algorithms to improve reliability in content-based pubsub systems and propose three algorithms based on different gossip strategies.

Unlike topic-based pubsub where topic subscriptions define a group of receivers and messages within a topic can be assigned a sequence number to easily detect losses, patterns

in content-based pubsub do not have the same effect. This is mainly linked to the fact that events can match multiple patterns resulting in routing being performed independently and messages not getting assigned a sequence number. The authors identify the previous problems as the main challenges that have to be addressed when applying reliability to content-based pubsub

In gossip algorithms, when a node wants to propagate a message to every node, instead of broadcasting the message, it picks a subset of the nodes, from the set of nodes in the system, to send the message to and when each node receives the message, each repeats the process. There are two main types of gossip: *push* and *pull*. In *push* gossip, each node gossips periodically, to disseminate its view of the system to other processes, while in *pull* gossip, a node requests the trasmission of information from other nodes to compensate local losses.

The first proposed algorithm, uses a strategy based on *push* gossip. At each gossip round, the gossiper randomly picks a pattern p from its subsciption table, constructs a digest, a pair containing the source identifier and a sequence number associated top the source, of the identifiers of all the cached events matching p, builds a gossip message containing the digest, labels it with p and propagates the message. Afterwards, when a dispatcher receives a message labelled with p, it checks if it is subscribed to this pattern and if the indentifiers contained in the digest correspond to events already received by it. The identifiers of missed events are included in a request message sent to the gossiper, which replies by sending the corresponding events.

The second and third algorithms use a strategy based on *pull* gossip. The former is a *subscriber-based* algorithm in which, when a dispatcher detects a lost event it inserts the corresponding information in a buffer *Lost*. In the next gossip round, it chooses a pattern p among the ones associated with local subscriptions, selects the events in *Lost* related with it, and inserts the corresponding information in a digest attached to a new gossip message. The message is then labelled with p and routed similarly to the *push* solution. Afterwards, when a dispatcher receives the message, it checks its cache against the events requested by the gossiper and, if any are found, sends them back to it. The last solution is a *publisher-based* algorithm. It requires that event sources also cache their published events and that the address of each dispatcher on the path towards a subscriber is appended to the message. It behaves similarly to the previous algorithm, but instead routes gossip messages towards publishers instead of publishers. The messages are distinguished based on the event source rather than the pattern, and augmented with the information required to route back to the publisher.

2.3.3.2 Fault-Tolerant Reliable Delivery of Messages in Distributed Publish/Subscribe Systems

In [18], the authors present a gossip-based approach to reliable delivery of messages over a reliable-topic, that relies on a special kind of node, which manages a set

of reliable-topics, called *repository* node. The *repository* facilitates reliable delivery from multiple publishers to multiple subscribers. Managing a reliable-topic involves two key components:

1. The *repository* should facilitate the registration and de-registration of authorized entities for reliable communications; and

2. The *repository* must support error-corrections, retransmissions and recovery from failures.

Reliable delivery of messages also involves two key aspects:

1. The *repository* needs to ensure that messages published by the publisher, over a reliable-topic are stored exactly-once, without gaps and in order at the repository managing this reliable-topic; and

2. The *repository* has to compute the intended destinations and ensure the reliable delivery of the stored message to the computed destinations.

The authors found that the use of a *repository* node, to manage reliable-topics, would result in clients having to wait for the *repository* to recover in case of failures, therefore becoming a single point of failure. To address this issue, each reliable-topic is managed by a set of *repositories*, called a *repository bundle*. Every *repository* inside a *repository bundle*, must manage the same reliable-topics. This solution provides redundancy, high-availability and more fault-tolerance.

2.3.3.3 Reliable and Highly Available Distributed Publish/Subscribe Service

In [14], the authors present a system that uses a topology management and subscription propagation scheme that enables brokers to compute new forwarding paths when failures of nearby neighbors occur.

The systems relies on a tree-based overlay, in which brokers maintain a partial view of this tree which includes all brokers within distance *f+1*, where *f* is the maximum number of concurrent broker failures that the algorithm tolerates. Such information is stored locally by each broker in a data structure called the topology map.

Brokers also maintain a subscription table containing entries for subscriptions inserted into the system. Subscriptions contain a *from* field that points to another broker located *f+1* hops closer to the subscriber. In the event of failure of one or more neighbors, the broker uses this information and its topology map to reconnect the network topology, and forward publications over new paths towards matching subscribers.

Finally, the algorithm contains a recovery procedure used by failed brokers to re-enter the system. This procedure aims at restoring a broker's routing information according to the current state of the network and is divided in three phases: *synchronization*, *message forwarding* and *termination of recovery*.

2.3.3.4 MinAvg-kTCO

In the paper [7], the authors propose the *MivAvg-kTCO* problem parameterized by *k* to use the minimum number of edges to create a *k-topic-connected overlay (kTCO)* for pubsub systems.

A *topic connected overlay (TCO)* is a per-topic connected graph of nodes subscribed to the topic, where exists a path between each pair nodes without being directly connected. Using *TCOs* guarantees that nodes not interested in that topic, do not contribute to disseminate information on that topic. The downside of *TCOs* is that connectivity is not guaranteed even if a single node fails. To address this issue, the authors introduce *k-topic-connected-overlay (kTCO)* which guarantee connectivity if fewer than *k* nodes fail simultaneously for the same topic.

The authors conclude the paper by presenting two algorithms that solve the *MivAvg-kTCO* problem: GM*2* to build *2*TCOs and Harary-Per-Topic (HPT) to build *k*TCOs where *k* is greater than 2.

2.3.3.5 P2S

P2S [6], is a topic-based and crash-tolerant *Paxos*-based pubsub middleware based on Goxos, a *Paxos*-based State Machine Replication framework. Paxos is a fault-tolerant consensus protocol in which a set of replicas tries to reach agreement on a value. With this protocol, replicas can reach agreement when at least *f+1* replicas are able to communicate, where *f* is the number of failures the system can tolerate.

This system leverages Paxos as a way to achieve total message ordering, while providing a mechanism to tolerate broker failures. For that purpose, P2S relies on a set of *2f+1* brokers between publishers and subscribers. With this design, P2S is able to provide fault-tolerance through replication while providing total ordering through Paxos.

In this system, client messages are handled by the Goxos framework. When clients send messages, Goxos treats them as Paxos requests, orders and delivers them to the broker application layer, which forwards the messages to the subscribers according to the message topic.

2.4 Example glossary, acronyms, and symbols

This is the first occurrence of an abbreviation: abbreviation of a longer text (abbrev). And now the second occurrence of the same abbreviation: abbrev. And a new acronym with capital letter: And extension of a xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto (xpto) and reused xpto. Let's also use a few other acronyms such as acornym aaa (aaa), acornym aab (aab), acornym aba (aba), acornym bbb (bbb) and xpto. In geometry, the area enclosed by a circle of radius r is πr^2. Here the Greek letter π is equal to the ratio of the circumference of any circle to its

Table 2.2: Solution properties.

System	Topology	Delivery	Ordering	Fault-tolerance	Strategy	Details	Requirements
2.3.3.3	Tree	Exactly-once	FIFO	✓	Redundant Edges	Partial view	Each broker maintains in its partial view all replicas within δ+1 distance, where δ is the number of faults to tolerate
P2S [6]	Star	X	Total	✓	Paxos	Replicates broker	2f+1 replicas, where f is the number of faults to tolerate

diameter. Lets add "computer" to the glossary! Be carefull with mathematical symbols in acronyms, please see the definition of μ.

3

A Short LaTeX Tutorial with Examples

This Chapter aims at exemplifying how to do common stuff with LaTeX. We also show some stuff which is not that common! ;)

Please, use these examples as a starting point, but you should always consider using the *Big Oracle* (aka, Google, your best friend) to search for additional information or al-ternative ways for achieving similar results.

3.1 Document Structure

3.2 Dealing with Bibliography

Citing something online.

3.3 Inserting Tables

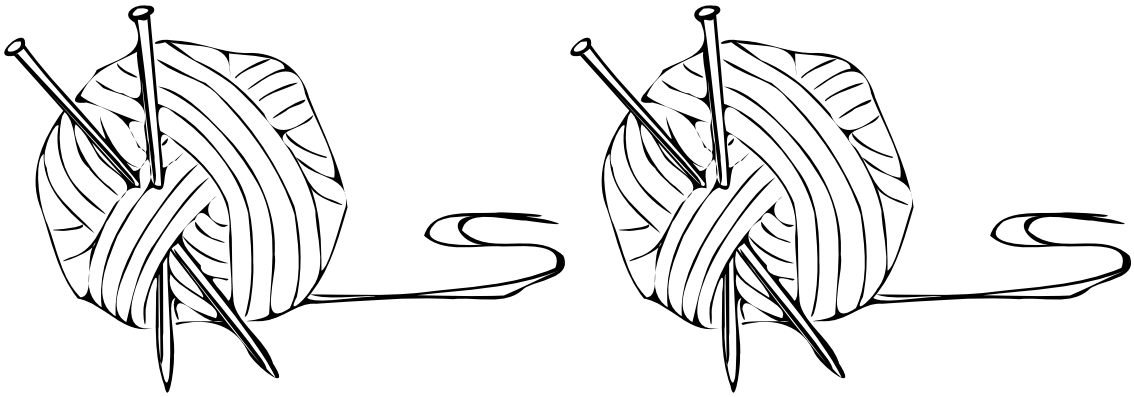
3.4 Importing Images

3.5 Floats, Figures and Captions

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.



(a) One sub-figure (b) Another sub-figure

Figure 3.1: A figure with two sub-figures!

And this is a small text that references the Figure 3.1 and its Subfigures 3.1(a) and 3.1(b).

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan

eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

3.6 Text Formatting

3.7 Generating PDFs from LaTeX

3.7.1 Generating PDFs with pdflatex

You may create PDF files either by using `latex` to generate a DVI file, and then use one of the many DVI-2-PDF converters, such as `dvipdfm`.

Alternatively, you may use `pdflatex`, which will immediately generate a PDF with no intermediate DVI or PS files. In some systems, such as Apple, PDF is already the default format for LaTeX. I strongly recommend you to use this approach, unless you have a very good argument to go for `latex` + `dvipdfm`.

A typical pass for a document with figures, cross-references and a bibliography would be:

```
$ pdflatex template
$ bibtex template
$ pdflatex template
$ pdflatex template
```

You will notice that there is a new PDF file in the working directory called `template.pdf`. Simple :)

Please note that, to be sure all table of contents, cross-references and bibliographic citations are up-to-date, you must run `latex` once, then `bibtex`, and then `latex` twice.

3.7.2 Dealing with Images

You may process the same source files with both `latex` or `pdflatex`. But, if your text include images, you must be careful. `latex` and `pdflatex` accept images in different (exclusive) formats. For `latex` you may use EPS ou PS figures. For `pdflatex` you may use JPG, PNG or PDF figures. I strongly recommend you to use PDF figures in vectorial format (do not use bitmap images unless you have no other choice).

3.7.3 Creating Source Files Compatible with both latex and pdflatex

Do not include the extension of the file in the `\includegraphics` command. E.g., use
`\includegraphics{sonwman}`
and not
`\includegraphics{sonwman.eps}`.
If you use the first form, `latex` or `pdflatex` will add an appropriate file extension.

This means that, if you plan to use only `pdflatex`, you need only to keep (preferably) a PDF version of all the images. If you plan to use also `latex`, then you also need an EPS version of each image.

To be included in the sections above

Para fazer citações, deverá usar-se a chave da referência no ficheiro BibTeX. Se for uma única referência, usar um "~" para ligar o `\cite{...}` à palavra que o precede (...`referência`). Caso queira fazer múltiplas citações, deverá agrupá-las dentro de um úinico `\cite{...}`.

Note que o ficheiro de bibliografia pode ter tantas entradas quantas quiser. Apenas aquelas cuja chave seja referenciada no texto é que serão incluidas na listagem de bibliografia.

Footnotes[1] will be numbered and shown in the bottom of the page.

A Tabela 3.1 ilustra alguns conceitos importantes associados à contrução de tabelas:

i) Não usar linhas verticais;

ii) A legenda deve ficar por cima da tabela;

iii) Usar as macros `\toprule`, `\midrule` e `\bottomrule` para fazer a linha horizontal superior, interiores e inferior, respectivamente.

Table 3.1: Test results summary.

Test	Anomalies	Warnings	Correct	Categories	Missed
Connection [**Beckman08**]	2	2	1	*C*	1
Coordinates'03 [**Artho03**	1	4	1	*2B, 1C*	0
Local Variable [**Artho03**]	1	2	1	*A*	0
NASA [**Artho03**]	1	1	1	—	0
Coordinates'04 [**Artho04**]	1	4	1	*3C*	0
Buffer [**Artho04**]	0	7	0	*2A, 1B, 2C, 2D*	0
Double-Check [**Artho04**]	0	2	0	*1A, 1B*	0
StringBuffer [**Flanagan0**	1	0	0	—	1
Account [**Praun03**]	1	1	1	—	0
Jigsaw [**Praun03**]	1	2	1	*C*	0
Over-reporting [**Praun03**]	0	2	0	*1A, 1C*	0
Under-reporting [**Praun03**]	1	1	1	—	0
Allocate Vector [**IBM-Rep**]	1	2	1	*C*	0
Knight Moves [**Beckman08**]	1	3	1	*2B*	0

[1] This is a simple footnote.

Total	12	33	10	5A, 6B, 10C, 2D	2

As figuras a inserir no documento deverão ser de qualidade, preferencialmente em formato vectorial (PDF vectorial) e não em *bitmap* (PNG, JPG, etc). As imagens *bitmap* (Figura 3.2) não escalam bem e têm reflexos negatives na qualidade do seu documento. Pelo contrário, as imagens *vectoriais* Figura 3.3 escalam muito tanto quanto o necessário sem degradar a qualidade da imagem.

Só deve usar *screenshots* se não tive mesmo nenhuma alternativa. Em vez e gerar um *screenshot*, tente usar uma impressora virtual PDF e imprimir para um ficheiro PDF. Regra geral obterá um PDF vetorial. Mesmo que o seu PDF contenha imagens, elas terão sempre qualidade maior ou igual à que obteria com um *screenshot*.

Para agregar várias figuras numa única... Poderá assim referenciar o conjunto como Figura 3.4 ou as sub-figuras separadamente como 3.4() e 3.4(a).

3.8 Equações

O LaTeX é uma ferramenta poderosa para escrever em estilo matemático. Permite inserir fórmulas no meio do texto como por exemplo esta: $ax^2 + bx + c = 0$. Também permite que as fórmulas sejam destacadas numa linha separada e centradas na página

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

ou numeradas

$$aaa \tag{3.1}$$

que depois pode ser referida no texto como sendo a equação 3.1

$$aa$$

$$a \tag{3.2}$$

$$b \tag{3.3}$$

$$c \tag{3.4}$$

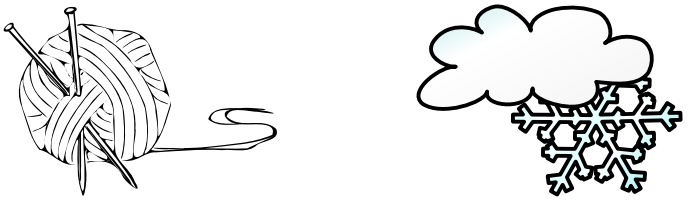
$$\tag{3.5}$$

3.9 Test for algorithms

Uncomment the algorithms source below and add the following to file "`5_packages.tex`"

Figure 3.2: Imagem em formato *bitmap* (JPG)

Figure 3.3: Imagem em formato PDF vectorial



(a) Novelo de lã (b) Tempestade com neve

Figure 3.4: Exemplo de utilização de *subbottom*

```
\usepackage{algorithm2e}
\RestyleAlgo{ruled}
```

and uncomment

`\ntaddlistof{listofalgorithms}`

in file "`8_list_og.tex`".

Bibliography

[1] H. N. S. Aldin et al. "Consistency models in distributed systems: A survey on definitions, disciplines, challenges and applications". In: *CoRR* abs/1902.03305 (2019). arXiv: 1902.03305. URL: http://arxiv.org/abs/1902.03305 (cit. on pp. 12, 13).

[2] D. Bermbach and J. Kuhlenkamp. "Consistency in Distributed Storage Systems - An Overview of Models, Metrics and Measurement Approaches". In: *Networked Systems - First International Conference, NETYS 2013, Marrakech, Morocco, May 2-4, 2013, Revised Selected Papers*. Ed. by V. Gramoli and R. Guerraoui. Vol. 7853. Lecture Notes in Computer Science. Springer, 2013, pp. 175–189. DOI: 10.1007/978-3-642-40148-0_13. URL: https://doi.org/10.1007/978-3-642-40148-0%5C_13 (cit. on p. 13).

[3] D. Bermbach, L. Zhao, and S. Sakr. "Towards Comprehensive Measurement of Consistency Guarantees for Cloud-Hosted Data Storage Services". In: *Performance Characterization and Benchmarking - 5th TPC Technology Conference, TPCTC 2013, Trento, Italy, August 26, 2013, Revised Selected Papers*. Ed. by R. Nambiar and M. Poess. Vol. 8391. Lecture Notes in Computer Science. Springer, 2013, pp. 32–47. DOI: 10.1007/978-3-319-04936-6_3. URL: https://doi.org/10.1007/978-3-319-04936-6%5C_3 (cit. on p. 12).

[4] S. Burckhardt. "Principles of Eventual Consistency". In: *Found. Trends Program. Lang.* 1.1-2 (2014), pp. 1–150. DOI: 10.1561/2500000011. URL: https://doi.org/10.1561/2500000011 (cit. on p. 11).

[5] M. Castro et al. "Scribe: a large-scale and decentralized application-level multicast infrastructure". In: *IEEE J. Sel. Areas Commun.* 20.8 (2002), pp. 1489–1499. DOI: 10.1109/JSAC.2002.803069. URL: https://doi.org/10.1109/JSAC.2002.803069 (cit. on p. 16).

[6] T. Chang et al. "P2S: a fault-tolerant publish/subscribe infrastructure". In: *The 8th ACM International Conference on Distributed Event-Based Systems, DEBS '14, Mumbai, India, May 26-29, 2014*. Ed. by U. Bellur and R. Kothari. ACM, 2014, pp. 189–197. DOI:

10.1145/2611286.2611305. URL: https://doi.org/10.1145/2611286.2611305 (cit. on pp. 20, 21).

[7] C. Chen, R. Vitenberg, and H. Jacobsen. "Brief announcement: constructing fault-tolerant overlay networks for topic-based publish/subscribe". In: *ACM Symposium on Principles of Distributed Computing, PODC '13, Montreal, QC, Canada, July 22-24, 2013*. Ed. by P. Fatourou and G. Taubenfeld. ACM, 2013, pp. 184–186. DOI: 10.1145/2484239.2484282. URL: https://doi.org/10.1145/2484239.2484282 (cit. on p. 20).

[8] P. Costa et al. "Epidemic Algorithms for Reliable Content-Based Publish-Subscribe: An Evaluation". In: *24th International Conference on Distributed Computing Systems (ICDCS) 2004, 24-26 March 2004, Hachioji, Tokyo, Japan*. IEEE Computer Society, 2004, pp. 552–561. DOI: 10.1109/ICDCS.2004.1281622. URL: https://doi.org/10.1109/ICDCS.2004.1281622 (cit. on p. 17).

[9] P. Costa et al. "Introducing reliability in content-based publish-subscribe through epidemic algorithms". In: *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems, DEBS 2003, Sunday, June 8th, 2003, San Diego, California, USA (in conjunction with (SIGMOD/PODS))*. Ed. by H. Jacobsen. ACM, 2003. DOI: 10.1145/966618.966629. URL: https://doi.org/10.1145/966618.966629 (cit. on p. 17).

[10] C. Esposito. "A tutorial on reliability in publish/subscribe services". In: *Proceedings of the Sixth ACM International Conference on Distributed Event-Based Systems, DEBS 2012, Berlin, Germany, July 16-20, 2012*. Ed. by F. Bry et al. ACM, 2012, pp. 399–406. DOI: 10.1145/2335484.2335537. URL: https://doi.org/10.1145/2335484.2335537 (cit. on p. 17).

[11] P. T. Eugster et al. "The many faces of publish/subscribe". In: *ACM Comput. Surv.* 35.2 (2003), pp. 114–131. DOI: 10.1145/857076.857078. URL: https://doi.org/10.1145/857076.857078 (cit. on pp. 14–16).

[12] GeeksForGeeks. *Strict consistency or Linearizability in System Design*. 2023. URL: https://www.geeksforgeeks.org/strict-consistency-or-linearizability-in-system-design/ (visited on 2023-10-04) (cit. on p. 12).

[13] P. Jalote. *Fault tolerance in distributed systems*. Prentice Hall, 1994. ISBN: 978-0-13-301367-2 (cit. on p. 16).

[14] R. S. Kazemzadeh and H. Jacobsen. "Reliable and Highly Available Distributed Publish/Subscribe Service". In: *28th IEEE Symposium on Reliable Distributed Systems (SRDS 2009), Niagara Falls, New York, USA, September 27-30, 2009*. IEEE Computer Society, 2009, pp. 41–50. DOI: 10.1109/SRDS.2009.32. URL: https://doi.org/10.1109/SRDS.2009.32 (cit. on p. 19).

[15] Y. Liu and V. Vlassov. "Replication in Distributed Storage Systems: State of the Art, Possible Directions, and Open Issues". In: *2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2013, Beijing, China, October 10-12, 2013*. IEEE Computer Society, 2013, pp. 225–232. DOI: 10.1109/CyberC.2013.44. URL: https://doi.org/10.1109/CyberC.2013.44 (cit. on p. 11).

[16] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. 2).

[17] G. Mühl. "Large-scale content based publish, subscribe systems". PhD thesis. Darmstadt University of Technology, Germany, 2002. URL: http://elib.tu-darmstadt.de/diss/000274 (cit. on p. 14).

[18] S. Pallickara, H. Bulut, and G. C. Fox. "Fault-Tolerant Reliable Delivery of Messages in Distributed Publish/Subscribe Systems". In: *Fourth International Conference on Autonomic Computing (ICAC'07), Jacksonville, Florida, USA, June 11-15, 2007*. IEEE Computer Society, 2007, p. 19. DOI: 10.1109/ICAC.2007.18. URL: https://doi.org/10.1109/ICAC.2007.18 (cit. on p. 18).

[19] A. L. P. F. de Sousa et al. "Partial Replication in the Database State Machine". In: *IEEE International Symposium on Network Computing and Applications (NCA 2001), October 8-10, 2001, Cambridge, MA, USA*. IEEE Computer Society, 2001, pp. 298–309. DOI: 10.1109/NCA.2001.962546. URL: https://doi.org/10.1109/NCA.2001.962546 (cit. on p. 11).

[20] A. S. Tanenbaum and M. van Steen. *Distributed systems - principles and paradigms, 2nd Edition*. Pearson Education, 2007. ISBN: 978-0-13-239227-3 (cit. on pp. 12–14).

[21] S. Tarkoma. *Publish/subscribe systems: design and principles*. John Wiley & Sons, 2012 (cit. on p. 16).

A

NOVAthesis covers showcase

This Appendix shows examples of covers for some of the supported Schools. When the Schools have very similar covers (e.g., all the schools from Universidade do Minho), just one cover is shown. If the covers for MSc dissertations and PhD thesis are considerable different (e.g., for FCT-NOVA and UMinho), then both are shown.

B

Appendix 2 Lorem Ipsum

This is a test with citing something in the appendix.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea

dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

I

Annex 1 Lorem Ipsum

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum

wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.