

# Distributed Systems

Name : Manel\_Martínez\_Díaz

Group : 288373011

1	Introduction.....	1
2	Implementation.....	1
3	Tests.....	8
3.1	Phase 2 .....	10
3.2	Phase 3 .....	11
3.3	Phase 4.1 .....	12

## 1 Introduction

I have implemented different methods that are described at the chapter 2.

The methods implemented are in the java classes:

- Log: class that logs operations
- TimestampVector: class to maintain the summary
- TimestampMatrix: class to maintain the acknowledgment matrix of timestamps
- TSAESessionOriginatorSide: Originator protocol for TSAE.
- TSAESessionPartnerSide: Partner's protocol for TSAE.
- ServerData: contains Server's data structures required by the TSAE protocol ( log , summary , ack ) and the application ( recipes).

I tested locally and in DS LAB, the results are described in detail on chapter 3.

At the end I conclude with some ideas about the protocol, this practical assignment and what we can do.

## 2 Implementation

I describe all methods implemented in this project.

### Method Clone on Class TimestampVector.

I override the clone method and because of the constructor TimestampVector only accept List, I get the size of the actual timestampVector and create a table of the same size. Then I initialize a new TimestampVector using the table converted as a List.

```
String[]participants =timestampVector.keySet().toArray(new
String[timestampVector.keySet().size()]);
TimestampVector timeclone = new
TimestampVector(Arrays.asList(participants));
```

After that I iterate the timestampVector and for each entry I call updateTimestamp for the TimestampVector copy the timestamp.

```
timeclone.updateTimestamp(thisTimestamp);
```

### Method Clone on TimestampMatrix Class:

I override the actual clone method and I instantiated a new TimestampMatrix to use it as a copy. To copy all the objects inside timestampMatrix I iterate the object and for each entry of the local timestampMatrix I call put method to add to the TimestampMatrix to copy from the TimestampVector, the key and a clone of the TimestampVector.

### Method updateTimestamp on class TimestampVector:

It is used for update a timestamp calling put method giving a timestamp. The method get the hostid of the timestamp to know which hostid is about to update and calling put method add the hostid and the timestamp updates the timestampVector.

```
this.timestampVector.put(timestamp.getHostid(), timestamp);
```

This method is useful to copy an object TimestampVector.

### Method Equals on Log, TimestampVector and TimestampMatrix Class:

On both classes I override the method equals and modify it to match our objects. First I check if the object is null, if is instance of the same class. If both are true then I create another Object and initialize to the Log/TimestampVector/TimestampMatrix passed in the method.

```
Log other = (Log) obj;
TimestampVector other = (TimestampVector) tsVector;
TimestampMatrix other = (TimestampMatrix) obj;
```

Then I called again the method equals to compare the Log/TimestampVector with the auxiliary one that I created.

```
return this.Log.equals(other.Log);
return this.timestampVector.equals(other.timestampVector);
```

But in the TimestampMatrix y need to iterate each key of the object and compare.

```
for (String name: this.timestampMatrix.keySet()) {
return
this.timestampMatrix.get(name).equals(other.timestampMatrix.get(name));
}
```

### Method addRecipe on ServerData Class:

A Timestamp object is created and also a new recipe, then I initialize an new Operation object calling AddOperation using as parameters the objects creates, the timestamp and the recipe created.

```
Timestamp timestamp= nextTimestamp();
Recipe rcpe = new Recipe(recipeTitle, recipe, groupId,
timestamp);
Operation op=new AddOperation(rcpe, timestamp);
```

After that, first we call method add from to add the operation to the log, second call method updateTimestamp to update the summary with the timestamp created and finally we call method add to add the recipe to the recipes.

```
this.log.add(op);
this.summary.updateTimestamp(timestamp);
this.recipes.add(rcpe);
```

### Method add on Log Class:

Is used to insert an operation into the log.

Get the timestamp of the operation, and the hostid of that timestamp.

```
Timestamp timestamp = op.getTimestamp();
String hostId = timestamp.getHostid();
```

Iterate the operations of the log of the hostid of the timestamp and search if the timestamp of each operation, then if the timestamp of the operation to add is equal or newer the operation is added to the log.

### Metodo updateMax on class:

Iterate each key of the timestampVector giving and get the timestamp of the last node received.

```
for (String node : this.timestampVector.keySet()) {
    Timestamp otherTimestamp = tsVector.getLast(node);
```

For each key compare the timestamp of the timestampVector and the actual timestampVector and if the local timestampVector is earlier is updated with the new timestamp.

```
this.timestampVector.replace(node, otherTimestamp);
```

### Method listNewer on Log class:

Create a List of operations and iterate the log. For each key get the operation from log and the last timestamp of the summary.

```

List<Operation> missingList = new Vector<Operation>();

for (String node : this.Log.keySet()) {
    List<Operation> operations = this.Log.get(node);
    Timestamp timestampToCompare = sum.getLast(node);

```

For each operation compare the timestamp operation and the timestamp of the summary and if the timestamp operation is newer is added to the operation list, that will be sent to the other peer at the entropy exchange.

```

for (Operation op : operations) {
    if (op.getTimestamp().compare(timestampToCompare) > 0) {
        missingList.add(op);
    }
}

```

### About synchronization

Using the scheme give in that practice, I briefly describe how to manage the communications between peers, using some synchronization mechanism to avoid interferences.

### Class OriginatorSide:

- **Open a socket** to communicate.
- Get a **copy** of the actual **ACK** and **Summary**
- Do:
  - o A **deep copy** of local **summary**, calling clone method.
  - o **Update** the **ACK** with update method from TimestampMatrix class.
  - o A **deep copy** of the local **ACK**, calling clone method.
- **Send** the **local Summary** and **local ACK**.

PARTNER SIDE RECEIVE AND PROCESS INFO  
 ORIGINATOR SIDE WAITS INFO FROM PARTNER SIDE

- Listen from partner to **receive messages**:
  - o **Operation** messages are **added** to a **List**
  - o **Wait** a Request message from **partner** with **Summary**
- Prepare operations to send:
  - o Identify **operations** contained **in the log but not in the local summary**. Iterate all objects of operations list and call method listNewer.
  - o **Send the operations** identified
  - o **Send** and "**end of TSAE session**" message
  - o **Receive message** to inform about the ending of the **TSAE session**.
- **Iterate operations** and call method execOperation to **add/remove operation** to update log and recipe to update recipes.
  - o If operation is an **ADD type**, call execOperation with a type **AddOperation** parameter.
  - o Otherwise is a remove operations and **call execOperation with a type Re-moveOperation** parameter.
- **Update summary, ACK and purge Log**.
- **Close the socket** connection.

### Class PartnerSide:

- Open a socket to communicate.
- Listen from originator to **receive messages**:
  - o **Wait a Request message from Originator with Summary an ACK**
  - o **Wait a Request message from originator with Summary**
- Get a **copy** of the actual **ACK** and **Summary**
- Do:
  - o A **deep copy** of local **summary**, calling clone method.
  - o **Update** the **ACK** with update method from TimestampMatrix class.
  - o A **deep copy** of the local **ACK**, calling clone method.
- Prepare operations to send:
  - o Identify **operations** contained **in the log but not in the local summary**. Iterate all objects of operations list and call method listNewer.
  - o **Send the operations** identified
- **Send the local Summary and local ACK.**

ORIGINATOR SIDE RECEIVE AND PROCESS INFO

PARTNER SIDE WAITS INFO FROM ORIGINATOR SIDE

- Listen from partner to **receive messages**:
  - o **Operation** messages are **added** to a **List**
  - o **Wait a Request message from partner with Summary**
  - o **Receive message** to inform about the ending of the **TSAE session**.
  - o **Send** and "**end of TSAE session**" message
- **Iterate operations** and call method execOperation to **add/remove operation** to update log and recipe to update recipes.
  - o If operation is an **ADD type**, call execOperation with a type **AddOperation** parameter.
  - o Otherwise is a remove operations and **call execOperation with a type Re-moveOperation** parameter.
- **Update summary, ACK and purge Log.**
- **Close the socket** connection.

### Method mergeMin on class TimestampVector:

Iterate the timestampVector gived and get the node, the timestamp and the last timestamp.

```
for (Map.Entry<String, Timestamp> entry : tsVector.timestampVector.entrySet()) {  
    String node = entry.getKey();  
    Timestamp otherTimestamp = entry.getValue();  
    Timestamp thisTimestamp = this.getLast(node);
```

Inside the for, if timestamp is null call method put to update the actual timestampVector with the node and the last timestamp received, and if the timestamp of the iteration is fewer than the last timestamp known, call replace method and update the timestamp with the node and the last timestamp received. With this we replace all timestamp with the minimum timestamp.

```

        if (thisTimestamp == null) {
            this.timestampVector.put(node, otherTimestamp);
        } else if (otherTimestamp.compare(thisTimestamp) < 0) {
            this.timestampVector.replace(node, otherTimestamp);
        }
    }

```

### Method minTimestampVector on TimestampMatrix Class:

Create a TimestampVector to save the timestamp known by all participants for each value of the timestampMatrix.

```
TimestampVector minTsV = null;
```

Iterate each value of the timestampMatrix, at first round clone the timestampVector and the next rounds call method mergeMin of the timestampVector for that round.

```

    for (TimestampVector matrixVector : this.timestampMatrix.values()) {
        if (minTsV == null)
            minTsV = matrixVector.clone();
        else
            minTsV.mergeMin(matrixVector);
    }

```

### Method purgeLog on Log class:

Call method minTimestampVector for acksummary to initialize a TimestampVector with the timestamps known by all participants.

```
TimestampVector minTimestampVector = ack.minTimestampVector();
```

Iterate each object of the log, to get the participant, the operation list and the minTimestampVector of the actual participant.

```

    for (Entry<String, List<Operation>> entry : log.entrySet()) {
        String participant = entry.getKey();
        List<Operation> operations = entry.getValue();
        Timestamp lastTimestamp = minTimestampVector.getLast(participant);
    }

```

Inside the for, iterate each operation and if the timestamp is earlier, which means is known by all participants, then remove the operation.

```

        for (int i = operations.size() - 1; i >= 0; i--) {
            Operation op = operations.get(i);
            if (op.getTimestamp().compare(lastTimestamp) < 0) {
                operations.remove(i);
            }
        }
    }

```

### Method execOperation on class ServerData.

There is two method one with AddOperation argument and another with RemoveOperation.

Both call method add of the class log to add the object operation on the argument to add the operation on the Log.

```
if (this.Log.add(addOp)) {
```

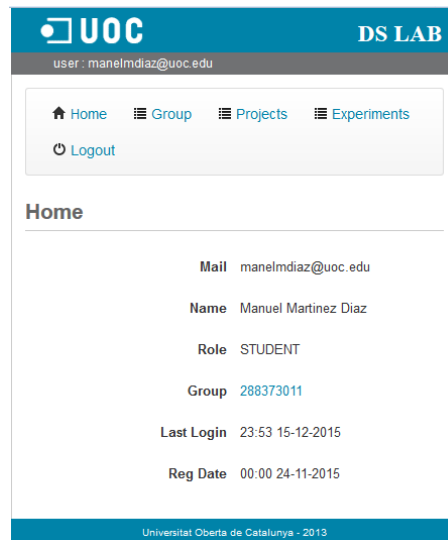
If the operation is done, then is call add or remove method from recipes to add the recipe or to remove the recipetitle.

```
    this.recipes.add(addOp.getRecipe());  
    this.recipes.remove(removeOp.getRecipeTitle());
```

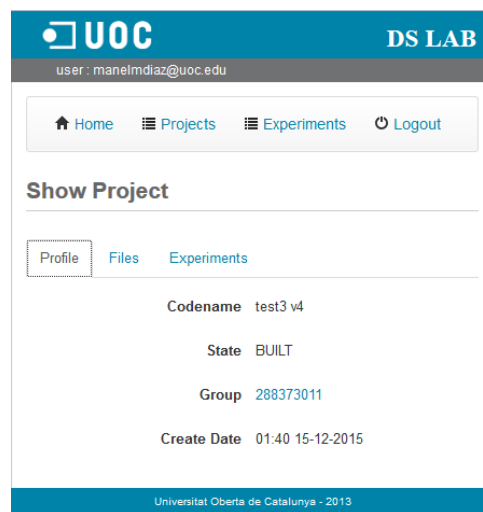
The method is synchronized to maintain the data integrity.

### 3 Tests

I use the DS LAB platform to test the project. I register and I have been assigned the **Group 288373011**.



I created several projects but the one to consider has the **Codename test3 v4**.



The project **'test3 v4'** has the classes indicated in the next image.



## Show Project

[Profile](#) [Files](#) [Experiments](#)

Name	Date	Size
<a href="#">TSAESessionOriginatorSide.java</a>	01:39 15-12-2015	6 KB
<a href="#">TimestampVector.java</a>	01:39 15-12-2015	7 KB
<a href="#">ServerData.java</a>	01:39 15-12-2015	7 KB
<a href="#">Log.java</a>	01:40 15-12-2015	5 KB
<a href="#">TimestampMatrix.java</a>	01:40 15-12-2015	4 KB
<a href="#">TSAESessionPartnerSide.java</a>	01:40 15-12-2015	6 KB

## Show Project

[Profile](#) [Files](#) [Experiments](#)

Id	State	Type	Phase	Nodes	Result Correct / Total
2.096	STARTED	DISTRIBUTED	3	0	1/1
2.212	STARTED	DISTRIBUTED	4.1	0	0
2.091	STARTED	DISTRIBUTED	3	0	2/3
2.207	STARTED	DISTRIBUTED	4.1	0	0/1
2.090	STARTED	DISTRIBUTED	3	0	0/2
2.099	STARTED	DISTRIBUTED	4.1	0	1/2
2.097	STARTED	DISTRIBUTED	3	0	2/3
2.095	FINISHED	DISTRIBUTED	2	0	2/2
2.249	STARTED	DISTRIBUTED	4.1	0	0
2.098	STARTED	DISTRIBUTED	4.1	0	1/2

During tests I have found different kind of problems and I want to explain briefly.

1. When I tried to build a project, DSLAB throws a parse error on the TSAESessionPartnerSide.java Class.

*"Parse error at TSAESessionOriginatorSide.java Encountered " "<" "<" "" at line 133, column 74. Was expecting one of: "(" ... "(" ..."*

To save all operations received in a list I use:

```
List<MessageOperation> operations = new ArrayList<>();
```


**Solution:**

I declare also specify the object type but the error remains.

```
List<MessageOperation> operations = new ArrayList< MessageOperation >();
```

At the end, I change to:

```
List<MessageOperation> operations = new Vector<MessageOperation>();
```

2. Sometimes after an experiment is launched the estate remains STARTED but the result only show a symbol .

**Solution:**

I created another experiment and launched again.

3. In the same experiment, after one is finished I Executed again but at the second or third time although it shows a result, the Estate remains STARTED and there is no possible to Execute again.

**Solution:**

I created another experiment and launched again.

## 3.1 Phase 2

**Local test:**

In linux command line I run:

```
manelmdz@ubuntu:~/Dropbox/UOC/Sistemes_Distribuits/2015-2016/2015t-practica-SD--students/2015t-practiques-SD--baseCode/scripts$ ./start.sh 20004 15 --logResults -path ../results --nopurge --noremove
```

The result log is saved in folder /sol/Phase2 - group288373011.data

**Result: Results are equal.**

```

group288373011@127.0.1.1:35010: 1

ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals

Results are equal      Nodes converged at the iteration 0

=====

***** num received results: 8
***** % received results: 53
***** minimal required number of results: 8

panelmdz@ubuntu:~/Dropbox/UOC/Sistemes_Distribuits/2015-2016/2015t-practica-SD--s
tudents/2015t-practiques-SD--baseCode/scripts$ █

```

### DSLAb test phase2:

In DS LAB I have made 1 experiment with two executions, both of them results OK.

Id	State	Type	Phase	Nodes	Result Correct/Total
2.095	FINISHED	DISTRIBUTED	2	0	2/2

## 3.2 Phase 3

### Local test:

In linux command line I run: `./start.sh 20004 15 --logResults -path ../results --noremove`

The result log is saved in folder `/sol/Phase3 - group288373011.data`

**Result: Results are equal.**

```

group288373011@127.0.1.1:35010: 4

ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals

Results are equal      Nodes converged at the iteration 0

=====

***** num received results: 8
***** % received results: 53
***** minimal required number of results: 8

manelmdz@ubuntu:~/Dropbox/UOC/Sistemas_Distribuits/2015-2016/2015t-practica-SD--s
tudents/2015t-practiques-SD--baseCode/scripts$ █

```

### DSLAb test phase3:

In DS LAB I have made 4 experiments with two or three executions each one. I noticed that the first one experiment fails but the rest of them have correct results.

Id	State	Type	Phase	Nodes	Result Correct/Total
2.090	STARTED	DISTRIBUTED	3	0	0/2
2.096	STARTED	DISTRIBUTED	3	0	1/2
2.097	STARTED	DISTRIBUTED	3	0	2/3
2.091	STARTED	DISTRIBUTED	3	0	2/3

## 3.3 Phase 4.1

### Local test:

In linux command line I run: `./start.sh 20000 15 --logResults -path ../results`

The result log is saved in folder `/sol/Phase4 - group288373011.data`

**Result: Results are equal, but not the logs.**

```
group288373011@127.0.1.1:35002: 2

ServerResult --- equals: logs are not equals
ServerResult --- equals: logs are not equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: logs are not equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: logs are not equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals
ServerResult --- equals: recipes are equals

Results are equal          Nodes converged at the iteration 4

=====

***** num received results: 3
***** % received results: 60
***** minimal required number of results: 3

manelmdz@ubuntu:~/Dropbox/UOC/Sistemes_Distribuïts/2015-2016/2015t-practica-SD--s
tudents/2015t-practiques-SD--baseCode/scripts$
```

**DSLAb test phase4:**

Id	State	Type	Phase	Nodes	Result Correct/Total
2.207	STARTED	DISTRIBUTED	4.1	0	0/1
2.099	STARTED	DISTRIBUTED	4.1	0	1/2
2.212	STARTED	DISTRIBUTED	4.1	0	0
2.249	STARTED	DISTRIBUTED	4.1	0	0
2.098	STARTED	DISTRIBUTED	4.1	0	1/2

## 4. Conclusions

It is difficult to understand the protocol TSAE and implement all the methods necessary to work fine but now I have a better understanding of the complexity of distributed systems and the protocol to communicate and maintain data integrity and availability.

I also notice that the test in DS LAB crash sometimes or even fails when in local works.

I consider the time and experience well invested to program in Java and comprehend better the theory, but only as a first approach and I think necessary to focus more in the synchronization of data and other protocols or even a simply version of other distributed protocol to ensure, the understanding the basics of how works a protocol and how to do a Java application implemented on Java .