Platform for Control and Delivery of services in Next Generation Networks

DELIVERABLE 4.2

Performance tests

Questions about this document or the features it describes should be directed to:

CEFRIEL
Via Fucini, 2
20133 Milano (MI)
Italy

Politecnico di Torino
Corso Duca degli Abruzzi, 24
10129 Torino (To)
Italy

Abstract

The PICO project concentrates on application streaming and context awareness as topical techniques (described in D1.2) to build distributed applications in the domain of emergency situations.

This document defines performance tests in order to evaluate the platform performance on both the server and the client side.

CHANGE LOG

Version	Date	Description
1.0	01/02/2011	First draft of the Deliverable 4
2.0	28/04/2011	Final version

Table of contents

Abstract **3**
Table of contents **5**
Introduction **7**
1. Test scenarios description **8**
1.1 Stress testing scenario 8
1.2 Endurance testing scenario 8
1.3 Car accident scenario testing 9
2. Methodology **11**
2.1 Stress test: PICO SERVER 11
2.2 Stress test: PICO CLIENT 12
2.3 Endurance test: PICO SERVER 13
2.4 Endurance test: PICO CLIENT 14
2.5 Car accident: PICO SERVER and PICO CLIENT interaction 15
3. Results **16**
3.1 Stress test results 16
3.1.1 Server side - Memory Usage 16
3.1.2 Client side – Context Updates received 17
3.2 Endurance test results 18
3.2.1 Server side - Memory Usage 18
3.2.2 Client side - Context Updates received 19
3.3 Car Accident Scenario test results 20
3.3.1 Server side - Memory Usage 20
3.3.2 Client side – Context Updates received 20
4. Evaluation and Conclusions **21**
Figures Index **22**
Bibliography **23**

Introduction

Performance testing is a subset of Performance engineering, a computer science practice that strives to build performance into the design and architecture of a system, prior to the onset of actual coding effort.

Performance engineering within systems engineering, encompasses the set of roles, skills, activities, practices, tools, and deliverables applied at every phase of the Systems Development Life Cycle which ensures that a solution will be designed, implemented, and operationally supported to meet the non-functional performance requirements defined for the solution.

Performance tests [1] are all of the activities involved in the evaluation of how the system can be expected to perform in the field. This is considered from a user's perspective and is typically assessed in terms of throughput, stimulus-response time, or some combination of the two. An important issue to consider when doing performance testing is scalability: the ability of the system to handle significantly heavier workloads than are currently required. This necessity might be due to such things as an increase in the customer base or an increase in the system's functionality. Either of these two changes would typically cause the system to have to be able to provide a significantly increased throughput. If there has not been appropriate testing to assure that the system can be scaled, unacceptable levels of denial of service or unacceptable response times might occur as workloads increase. This is likely to have a very negative impact on customer satisfaction and therefore retention.

In the first paragraph we give a brief introduction regarding the type of performance testing which will be run. In the second we will concentrate on the methodology related to the context of this project. In the third we will present the results and, finally, in the fourth one we will discuss the results and we will evaluate the platform.

1. Test scenarios description

In order to get a complete view of the PICO platform's behavior, different types of test have been performed: stress testing, endurance testing, scenario testing. In this section we describe the test scenarios defined.

1.1 Stress testing scenario

In software testing, a system stress test refers to tests that put a greater emphasis on robustness, availability, and error handling under a heavy load, rather than on what would be considered correct behavior under normal circumstances. In particular, the goals of such tests may be to ensure the software does not crash in conditions of insufficient computational resources (such as memory or disk space), unusually high concurrency, or denial of service attacks.

This is the *stress scenario*:

3 crises will be loaded into the DB.

Each crisis needs 300 users

900 users will be loaded into the DB (300 per type).

Through a IMS Java client we simulate a 900 users connection and 4300 context updates, each every 50 milliseconds.

1.2 Endurance testing scenario

Endurance testing is usually done to determine if the application can sustain the continuous expected load. During endurance tests, memory utilization is monitored to detect potential leaks. Also important, but often overlooked is performance degradation. That is, to ensure that the throughput and/or response times after some long period of sustained activity are as good or better than at the beginning of the test.

This is the *endurance scenario*:

3 crises will be loaded into the DB (not close each other).

Each crisis needs 33 users (2 per user type)

99 users will be loaded into the DB (33 per type).

Through an instrumented Java IMS client we simulate a 99 users connections and 150 context updates, each every 5 seconds.

1.3 Car accident scenario testing

This is a simple use case adapted to the final scenario.

We will instrument the PC, which is running the PICO server, and clients, which are running the PICO client. After that we will measure the time spent in order to perform these common operations as defined in the final scenario.

This is the car accident scenario:

2 crises will be loaded into the DB.

Each crisis needs 3 users (1 per user type)

6 users will be loaded into the DB (2 per user type).

After that these interactions will be performed:

- A car accident (A1) with three unconscious injured and 2 cars involved has just happened in the Zone 1 (Z1).
- A policeman (PSU1) reaches the Accident (A1).
- The main desktop of the policeman's device (PSCD1) offers the list of buddies and the position of the policeman in that moment. Among the list, the policeman selects the operator of the police station (OP1) and begins an audio call using a IMS Client.
- During the call, the operator (OP1) inserts via a web interface all initial data about the accident into PICO Server (S1) and share the application "First Emergency Call" (AP1) with him.
- In the notification area of the policeman's device the download status of the application (AP1) is shown, when it is downloaded, the policeman (PSU1) from the second desktop of the device (PSCD1) launches the first emergency call application (AP1) and starts scanning the QRCode of the first victim (V1)
- When QRCode is scanned, the application (AP1) shows a pop-up message with the Name and Last name of the victim (V1) and calls automatically the first number of an important person (e.g.. Parents) using the IMS Client. During the call the policeman warns the situation and read the name to the person.
- At the same time, the application using the information embedded in the QRCode, sends the name and other important information related to the victim (V1) to the nearest PICO Server (PICO1) PICO Server processes the information related to the accident (A1) and the information related to all users' contexts connected to the PICO Server. The reasoner of the PICO Server (PICO1) begins to analyze contexts and positions closer to the accident (A1). Reasoner will select the best PSCDUser available according also to other parameters such as: Team, kind of user....
- An ambulance (PSU2) with a paramedic is near the Zone 1(Z1), PICO Server (PICO1) via the reasoner alerts with a message the paramedic's device (PSCD2) that contains all medical information gained at 7) , using the embedded IMS Client, automatically connects with an audio call or video call (base on network condition, battery level), the paramedic(PSU2) with the policeman (PSU1). On the main desktop is shown a map with 2

IMS buddies, PSU1 and PSU2 and the best path to reach the Accident (A1) base on traffic conditions.

- On display of the PSCD2 a message with the availability of a diagnostic application (AP2) with some custom data related to the victim (V1) is shown. The paramedic (PSU2) is able to do download and install it during the route to the accident (A1).
- The policeman starts from the contextual application list(second desktop) an application to find the nearest tow truck to the zone 1 (Z2) and sends a sms with all accident details (location, number of cars)
- A squad of firefighters is in the Zone 1 (Z2)
- PICO Server starts the application that shows the best path to the FF and begins also a communication session between the Firefighter (PSCU3) and the policeman (PSCU1) based on network condition:
  - chat session (low traffic condition) (the first auto chat message contains all accident details)
  - Audio session (medium traffic condition)
  - Video session (high traffic condition)
- The policeman now has 2+1 active buddies in his buddy list and based on session started on 13), decides to: share a picture using a IMS session (IMS Client) to the FF cause a small fire was started after the accident. So the FF can see how big is the fire and position of the cars.
- A notification message on the PSCD1 device alerts the policeman that the contextual application list (second desktop) has been updated, so the policeman can switch the desktop, selects the First Aid Application (or PDF file from third desktop) and download/launch it.

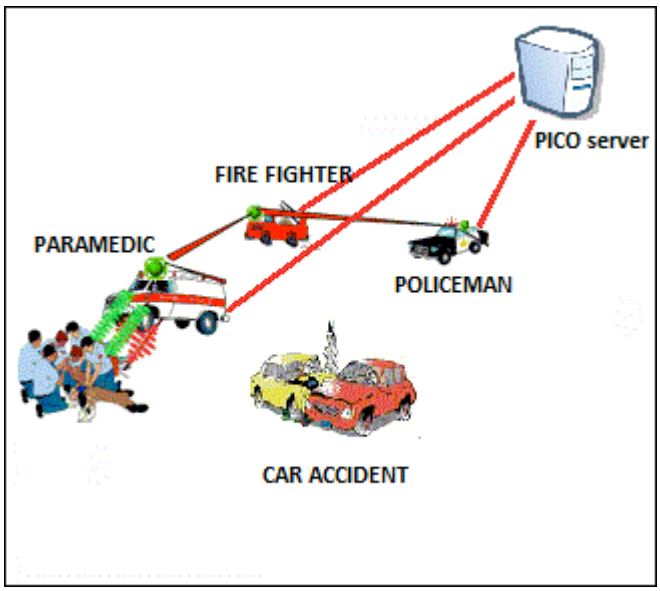

Figure 1 – Car Accident scenario.

2. Methodology

2.1 Stress test: PICO SERVER

For this kind of test we run the "PICO SERVER" on a virtual machine that has a new installation of Windows XP.

From another host we run an instrumented Java IMS client that simulates 900 user connections. This client sends to the "PICO Server" different ContextML files with predefined information in order to simulate different users connections to the server.

In this way the "PICO Server" reacts as well as 900 mobile devices tries to send information.

We also run a real mobile device in order to measure the number of update received.

For this kind of test will be measured:

- Virtual Machine Memory Usage
- Percentage of data loss
- Delay from USER n interaction with "PICO Server" to USER2 context update.

Figure 2 – A 300 users associated emergency

2.2 Stress test: PICO CLIENT

From the instrumented “PICO Server” will be sent to a real mobile device multiple updates (one every 50 milliseconds) in order to understand when the device is no longer able to manage them.

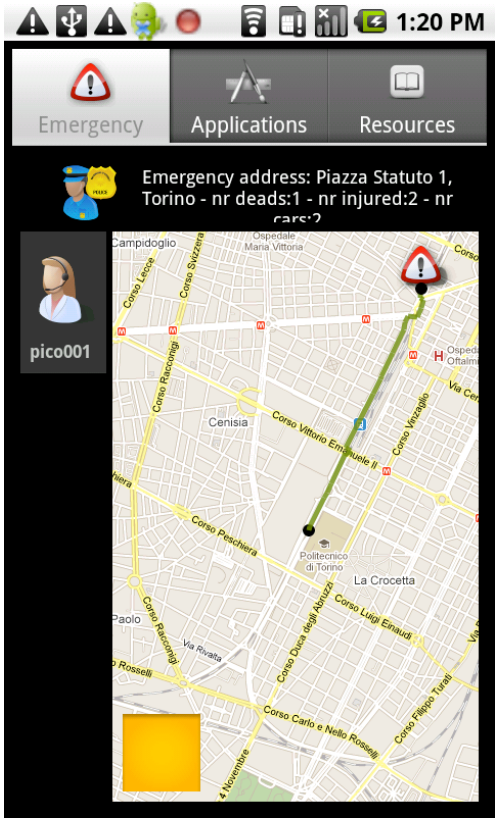

Figure 3 – An IMS error (red icon on the notification bar)

2.3 Endurance test: PICO SERVER

For this kind of test we run the “PICO SERVER” on a virtual machine that has a new installation of Windows XP.

From another we run an instrumented Java IMS client that simulates 100 user connections. This client sends to the “PICO Server” different ContextML files with predefined information in order to simulate different users connections to the server.

In this way the “PICO Server” reacts as well as 100 mobile devices tries to send information.

We also run a real mobile device in order to measure the number of update received.

For this kind of test will be measured:

- “PICO Server” Usage Memory
- Percentage of data loss
- Delay from USER n connection to USER2 context update.

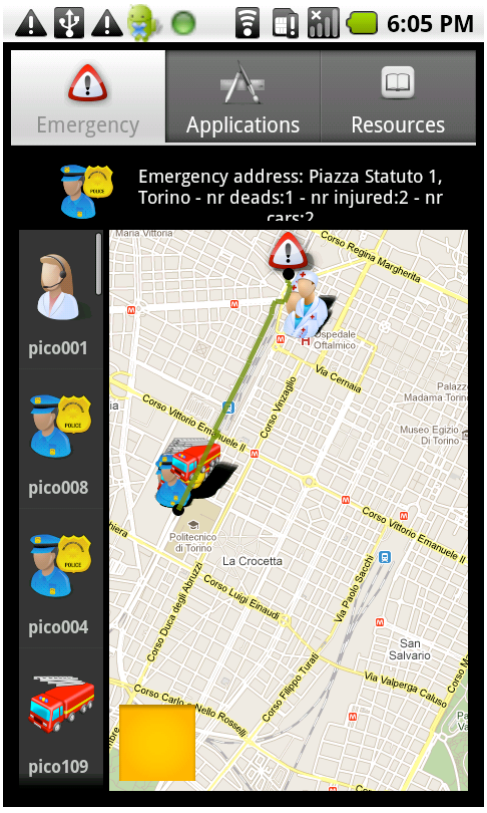

Figure 4 – A 33 users associated emergency

2.4 Endurance test: PICO CLIENT

From the instrumented "PICO Server" will be sent to a real mobile device 150 updates (one every 5 sec) in order to understand if the device is able to manage them.

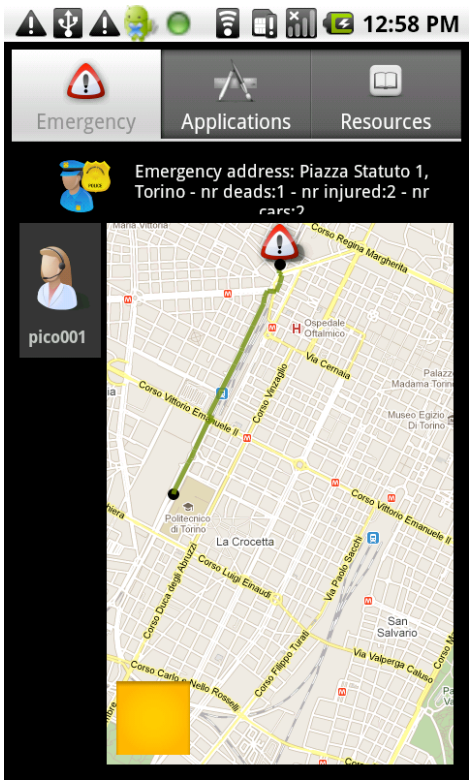

Figure 5 – A context update after an association with an emergency

2.5 Car accident: PICO SERVER and PICO CLIENT interaction

For this kind of test we will follow the scenario described in section 1.3.

Analyzing the operation, which has to be done by PICO SERVER and PICO CLIENTS, will be produced 32 notifications from PICO SERVER and all these operations are not critical because there are few users and few interaction.

We expect a 0% data loss and a minimum delay in the communication between server and clients.

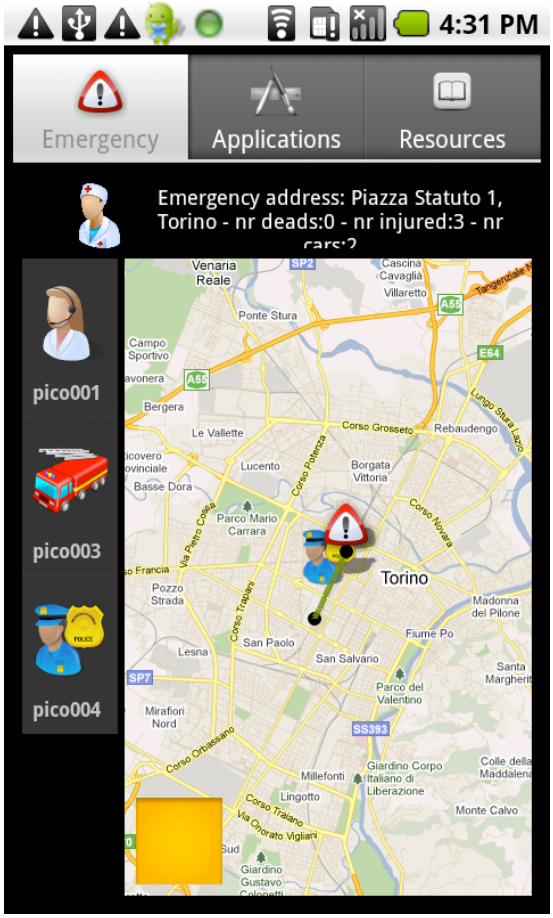

Figure 6 – Car accident scenario on Paramedic's device

3. Results

This section focuses on describing testing results. Each paragraph will show graphs, values, and finally, a brief description about results of the test in analysis.

3.1 Stress test results

3.1.1 Server side - Memory Usage

Memory usage of the pico server increases with the number of connectedusers.
Figure 7 represents the evolution of free memory in the system.
The creation of the three emergencies can bee easyly seen in the graph.
When the system had 900 user connected the memory occupation was 85 MB.

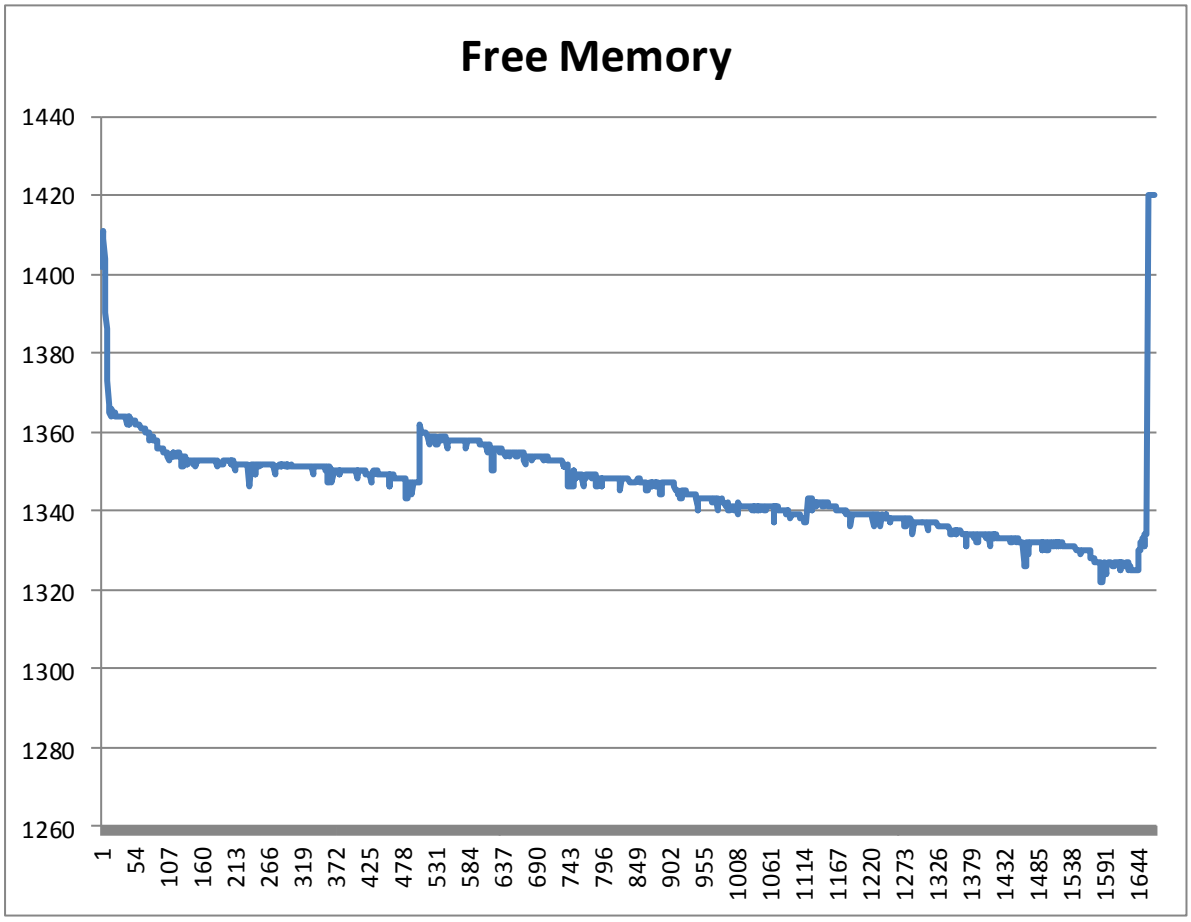

Figure 7 – Free memory trend in stress test scenario

3.1.2 Client side – Context Updates received

Number of updates sent by "PICO SERVER": 4302

Number of updates received by "PICO CLIENT": 179

Number of update per second: 20

Mean interval between two updates received by PICO CLIENT: 1208 milliseconds

St. Dev.: 2189 milliseconds

Data loss: 99,04%

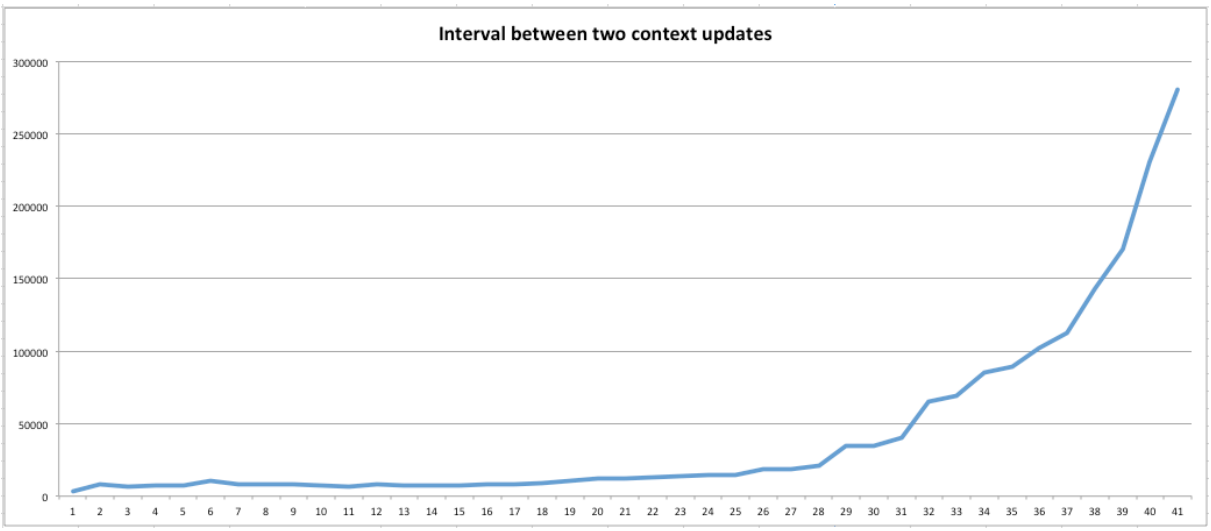

Figure 8 – Time between two context updates in stress test scenario

3.2 Endurance test results

3.2.1 Server side - Memory Usage

Even in this test, memory usage of the pico server increases with the number of connected users.
Figure 9 represents the evolution of free memory in the system.
Although the trend of available memory in the system is not clearly defined as the previous one, it is easy to see that the number of users connected to the system is directly proportional to the PICO Server memory usage.
In this case the number of user was 99 and the maximum memory occupation was 73 MB.

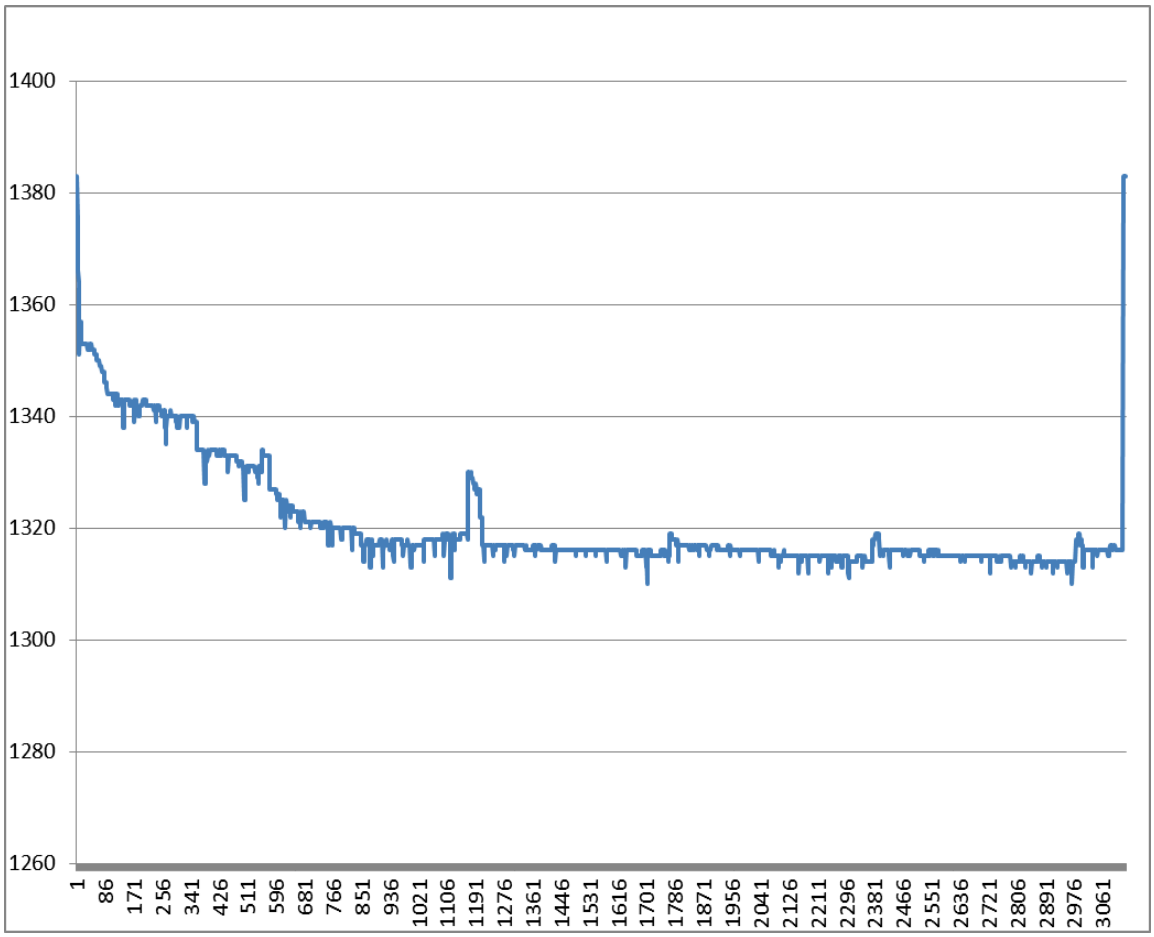

Figure 9 – Free memory trend in endurance test scenario

3.2.2 Client side - Context Updates received

Update sent by “PICO SERVER”: 100

Average of update received by “PICO CLIENT”: 33

Mean interval between two updates received by PICO CLIENT: 8695 milliseconds

St. Dev.: 1781 milliseconds

Average Percentage of data loss: 67%

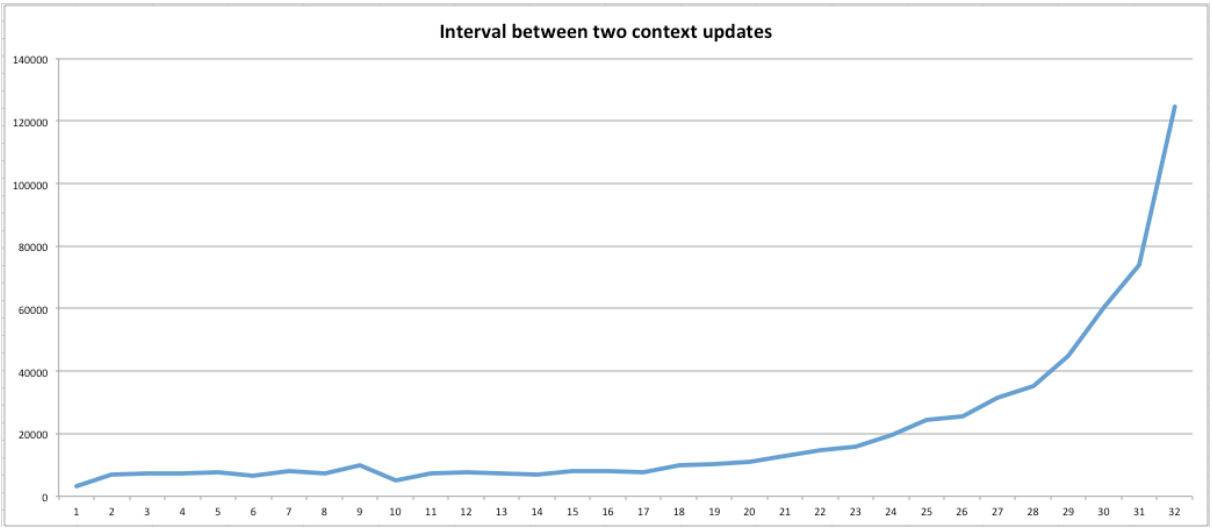

Figure 10 – Time between two context updates in endurance test scenario

3.3 Car Accident Scenario test results

3.3.1 Server side - Memory Usage

The maximum memory usage for this scenario is 66 MB.

3.3.2 Client side – Context Updates received

Update sent by “PICO SERVER”: 32

Updates received by “PICO CLIENT”: 32

Percentage of data loss: 0%

A graph that represents the interval between two context updates will not be shown because this metric depends by users interactions.

4. Evaluation and Conclusions

The data collected show that there is a decrease in performance of the platform when the number of users associated to a given emercency grows.
This is due to the quality of the implementation of the IMS client used (an open source project adapted to our purposes) which does not ensure the correct transmission of information (from server to client) if they occur too frequently.
Results show that the percentage of data loss increases dramaticaly in case of very frequent updates.
In case of many people associated with the same emergency, a single user context update causes a context update propagation to all the connected users.
For this reason the PICO platform is not immediately ready for a commercial use. A performant communication channel based on SIP /IMS is needed in order to improve the whole platform.
On the other hand the platform acted properly (with no data loss) in the use case scenario.
Furthermore there were no critical issues about PICO Server's memory management even in cases where many users were connected simultaneously and, finally, the Android clients had only side effect issues due to the IMS client.

Figures Index

Figure 1 – Car Accident scenario..... 10
Figure 2 – A 300 users associated emergency..... 11
Figure 3 – An IMS error (red icon on the notification bar) 12
Figure 4 – A 33 users associated emergency 13
Figure 5 – A context update after an association with an emergency 14
Figure 6 – Car accident scenario on Paramedic's device 15
Figure 7 – Free memory trend in stress test scenario 16
Figure 8 – Time between two context updates in stress test scenario..... 17
Figure 9 – Free memory trend in endurance test scenario 18
Figure 10 – Time between two context updates in endurance test scenario..... 19

Bibliography

[1] Avritzer, A., J. Kondek, D. Liu, and E. J. Weyuker, "Software Performance Testing Based on Workload Characterization," Proceedings of the 3rd International Workshop on Software and Performance, Rome, Italy, July, 2002, pp. 17-24

[2] http://softeng.polito.it/pico

[3] http://www.wikipedia.org/wiki/Software_performance_testing