ASPECTS OF DISTRIBUTED COMPUTING USING BLUETOOTH NETWORKS

Bogdan Ioan Fărtățescu

Teaching assistant, PhD. student, "Politehnica" University of Bucharest, Faculty of Control and Computers, Splaiul Independenții 313, Bucharest, Romania; fbogdani@cs.pub.ro

Abstract: This paper will focus on the possibility of distributed computing over a Bluetooth network. In the paper two experiments will be presented, one for a low computational application and one for a high computational application. Based on these experiments the paper will conclude on what requirements are in distributed computing over a Bluetooth network.

Key words: distributed computing, Bluetooth, mobile devices.

INTRODUCTION

As people become more and more mobile, so does the computer industry. The fact that mobile devices such as PDAs have become as powerful as some PCs creates new horizons for distributed computing to allow devices to benefit of higher computational capabilities than they can offer as a single. The Bluetooth technology (1) targeted the mobile computer, the mobile phone, small personal digital assistants and peripherals.

EXPERIMENTAL

First experiment was a relatively low computational on the distributed part. The experiment was made on a developed application called GameOfLife. The application involved a development of a matrix of so called individuals based on the following rules:

1. First stage involves a random generation of individual matrix with 0 symbolizing no individual and 1 symbolizing a living individual.
2. At every stage, the individuals are evolving; if around a living individual are too many neighbors the individual dies; if around a place with no individual are enough neighbors, a new individual appears there. The matrix from one generation is taken in account in all the evolving of the next generation, meaning that a change in next generation is visible only when the evolution ends.

After every new generation, the image of the matrix is shown on the phone's screen, and via infrared port is presented the time spent in both evolution and displaying of the matrix. The matrix sizes were 150 x 200.

The application was first implemented on a single device. Afterwards, it was re-implemented in a distributed way, meaning that the computational phase of the matrix was distributed between 1 or 2 clients. After initializing the matrix, the server sends the clients a part of the matrix to resolve. Each client compute a part of the matrix and then sends the data to the server. When all matrix is computed, the server draws the game and another evolving phase started. For optimizing purposes, the matrix was represented at bit level.

The second experiment was a high computational one application. The application computes the first prime number greater than a start value, in the experiment being used 12375143. The first prime number greater than 12375143 is 12375161.

The application starts searching for prime numbers from the start value and increments the number with a specified step at each new iteration. The application takes all odd numbers between 2 and half of the tested number. If none of these divides the number, then the number is prime and the searching stops.

As in previous experiment, the application was first implemented on a single device. For this implementation, the step is 2.

Afterwards, the application was re-implemented in a distributed way, the searching phase being distributed between 1 or 2 clients. After connecting with the clients, the servers sends the starting number and the step number. The server starts with the start number. For n clients, the starting number is:

*Client_start_number = Server_start_number + 2*client_number*;

All step numbers are equal: *Step_number = 2*(n+1)*.

After all clients have been given the start number and the step number, all clients and the server start searching for the prime number using the algorithm presented in the implementation on a single device. When a client finds a prime number, it stops execution and sends the server this number. The server then verifies if this prime number is smaller than the one he got before. If no prime number were got before or if this number is smaller, the server retains this as smallest prime found and sends it to all clients. If a client is searching for a prime number and gets a number from the server, it verifies if he is testing a value bigger than this number. If so, it stops execution. If not, he will continue searching until a prime number is found or the value tested becomes bigger than the number got from the server. This way, a client will always search for prime numbers only if no smaller prime numbers have been found. The same algorithm applies to the server too. The

execution stops when all clients and the server have finished searching, and the prime number the server have is the smallest.

RESULTS AND DISCUSSIONS

The results of the first experiment on single device and in distributed way are presented in table bellow. For these results the evolution time is the time between starting the evolution and the moment all clients have send their results and got the lines from the server. The time is measured in milliseconds.

	Time	1st evolution	2nd evolution	3rd evolution	4th evolution	5th evolution
Single	Evol.	2109	2216	2156	2056	2312
	Draw	1344	890	609	1105	928
	Total	3453	3106	2765	3161	3240
1 client	Evol.	1456	1537	1286	1482	1396
	Draw	1218	1190	856	922	836
	Total	2674	2727	2142	2404	2232
2 clients	Evol.	1236	1287	1156	1202	1243
	Draw	1126	956	920	1089	962
	Total	2362	2243	2076	2291	2205

The results are presented in figure 1 in a comparative manner. The performance gain is presented in figure 2. For 1 client, the medium performance gain is 22.47%. For 2 clients, it's 28.75%. The network bandwidth has a significant impact over the gain.

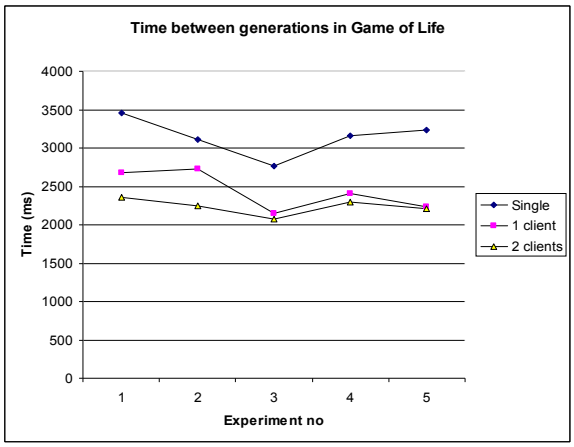

Figure 1

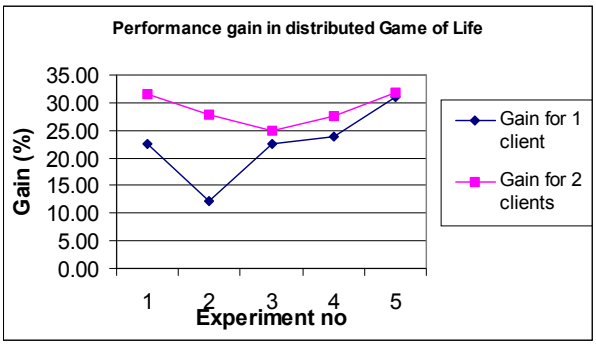

Figure 2

The result for the second experiment for 5 different runs of the application on a single device and in distributed way are presented in table bellow. For these results the time is the time between the moment the server begins sending start and step numbers and the moment all clients and server have finished searching. The time is measured in milliseconds.

	1st run	2nd run	3rd run	4th run	5th run
Single	21344	23302	20186	21615	20965
1 client	11286	10854	10487	12076	11147
2 clients	9268	8356	9028	7914	8456

The results are presented in figure 3 in a comparative manner. The performance gain is presented in figure 4. For 1 client, the medium performance gain is 47.05%. For 2 clients, it's 62.27%. The values are very close to the ideal values.

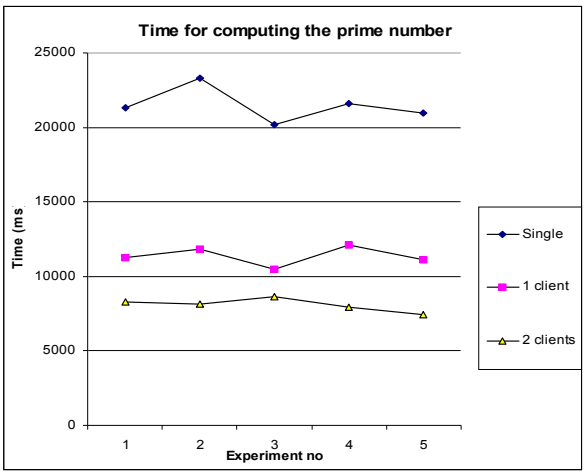

Figure 3

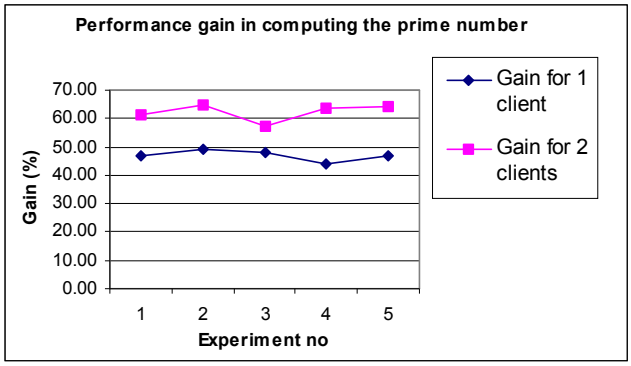

Figure 4

CONCLUSIONS

As seen in experiments presented above, a distributed application over a Bluetooth network benefits better of distributed execution when the difference between the amount of computed data and the amount of transmitted data is bigger. When separation has been made between low and high computational applications, was taken into account the rapport between the amount of time spent computing inside the replicated part and the total time spent over entire application (2). As a conclusion, to better benefit from the distributed computing environment, only high computing applications should be distributed, and also after careful selection of the part of code which is distributed and optimal communication.

REFERENCES

1. Palo Wireless; Bluetooth Resource Center documentation;
2. Fărtăţescu, B. I.; 2005; "Aspects of Grids Over Mobile Phone Network"; CSCS15; "Politehnica" University of Bucharest, Bucharest, Romania;