A real-time scheduling framework on distributed mobile environments

Un framework de planificación en tiempo real en entornos móviles distribuidos

Adriana Hernández Beristain[1], Mariano Larios-Gómez[1*], Mario Anzures García[1], Franco Rojas-López[2] y Erika A. Martínez-Miron[1]

[1]Facultad de Ciencias de la Computación
Benemérita Universidad Autónoma de Puebla-México
[2]Universidad Politécnica Metropolitana de Puebla
*mariano.larios@correo.buap.mx

PALABRAS CLAVE:

Tecnología Educativa, Sistemas de Tiempo Real, Ambientes Distribuidos, Sistema Embebido, Programación de Tareas

RESUMEN

A lo largo de los años, un modelo centralizado se ha utilizado ampliamente en todo tipo de aplicaciones informáticas, educativas y de nuevas tecnologías. Esta estructura de aplicación de sistema móvil distribuido divide tareas o cargas de trabajo entre el proveedor y el servicio solicitante. Este trabajo describe la implementación de una interfaz gráfica de usuario, denominada JPeer, para un software embebido; esto muestra el uso de una red P2P que permite a una supercomputadora la asignación de sus recursos de manera óptima entre los diferentes nodos conectados a ella. Los pares en este proyecto se representan como dispositivos móviles y con el uso de JNI (interfaz nativa de Java), con esto es posible comunicar pares creados en Java con pares creados en C++, en consecuencia, el paso de mensajes sería posible entre diferentes programaciones. lenguajes y sistemas operativos. Aplicamos varias redes P2P con múltiples pares en un nodo de LNS (laboratorio de supercomputación) en el sureste de México. La comprensión de algoritmos de sistemas distribuidos y de tiempo real puede representar una dificultad debido a la abstracción y dificultad de aprendizaje. Por su parte, la implementación del framework representa un entorno de sistema distribuido móvil, donde el usuario puede gestionar los nodos de forma sencilla, fácil y transparente, así como visualizar cómo cada nodo ejecuta sus procesos, se convierte en una herramienta muy útil y didáctica. Por otro lado, destacamos la necesidad de adaptar lenguajes con características nativas y aprovechar ambas partes en entornos educativos y tecnológicos.

KEYWORDS:

Educational Technology, Real-Time Systems, Distributed environments, Embedded System, Scheduling Tasks

ABSTRACT

Throughout the years, a centralized model has been widely used in all sorts of regarding computer science, educational and new technology applications, this distributed mobile system application structure partitions task or workloads between the provider and requester service. This work describes the implementation of a user graphical interface, named JPeer, for an embedded software; this shows the use of a P2P network that allows a supercomputer the allocation of its resources optimally among the different nodes connected to it. The peers in this project are represented as mobile devices and with the use of JNI (Java Native interface), with this it is possible to communicate peers created in Java with peers created in C++, accordingly, message passing would therefore be possible among different programming languages and operating systems. We applied several P2P nets with multiple peers in a node of LNS (supercomputing laboratory) in Southeast Mexico. The understanding of distributed and real time system algorithms can represent a difficulty due to the abstraction and difficult learning. In the meantime, the framework implementation represents a mobile distributed system environment, where the user can manage the nodes in a simple, easy and transparent way, as well as visualize how each node executes its processes, becomes a very useful and didactic tool. On the other hand, we highlight the need to adapt languages with native characteristics and take advantage of both parts on educational and technological environments.

• **Recibido: 23 de julio 2021** • **Aceptado: 10 de abril 2022** • **Publicado en línea: 28 de febrero 2023**

ProgMat,2023,15,1; https://progmat.uaem.mx/progmat

DOI: 10.30973/progmat/2023.15.1/6

1. INTRODUCTION

Real-time and distributed systems are included and used in the latest generation of aircraft, spacecraft, robot control, wireless mobile device (WMD) communication, educational technology, etc. In this environment. Real-time algorithms must be capable of immediate response because they have restrictions regarding the use of their limited resources. It was proposed to construct a routing algorithm to maintain and construct a task scheduler, considering an online compensation for the distribution of loads, in addition to optimizing the route of the messages, reducing the communication time based on the routing problem [1],[2],[3].

In a mobile distributed system (MDS), we have nodes (mobile devices) as a dynamical system, so it is necessary to perform the analysis of the metric to be used, which is the minimization of the sum of the weights measured at determined times, based at a cost of sending a packet between two nodes. The metric is important since in the WMD, the difference of values is imparted on the complemented system [4]. This mentioned metric is based on [5], where it compares different algorithms based on theorems of process planners on uniprocessors and multiprocessors by measuring the computation time required for the determination of a scheduler that satisfies partial order and resource constraints.

In an MDS, one of the important elements is the metric of time and resources in the computation cost. Also, there is an important element hybrid as a single entity with three elements: sending node, receiving node, bridge node or router node. It can be viewed as a local state, depending on the requirements and constraints to make the communication. In an MDS, the nodes can communicate with neighboring nodes by means of an appropriate entity configuration, in a transmission range from a sender to a receiver in a L neighborhood. Communication in this MDS configuration neighborhood describes a dynamic topology, which makes it difficult to establish and maintain a communication path between the nodes $v_i \rightarrow v$j, considering that $i \neq j$ and v_i, $v_j \in V$, where V is the set of nodes in a mobile network, presumably remote because of the unwillingness to communicate in each time.

Routing algorithms in an MDS environment are based on message flooding, such as a route search mechanism in an L neighborhood, sending messages in neighborhoods of nodes, using Ad Hoc sustainable longevity routing (SLR) networks [6], [7].

Peer-to-peer (P2P) paradigm is a classical P2P network, whereas all participating computers have equivalent capabilities and responsibilities. The nodes can directly exchange resources and services between each other without the need for centralized servers. They can collaborate to perform tasks by aggregating the pool of resources (e.g., storage, CPU cycles) available in the P2P network.

This work describes the implementation of a graphical interface, named JPeer, for an embedded software; this shows the use of a P2P network that allows to a supercomputer the allocation of its resources optimally among the different nodes connected to it. Sometimes, the understanding of distributed and real time system algorithms can represent a difficulty due to the associated abstraction. So, the implementation of JPeer Framework to represent a mobile distributed system environment, where the user can manage the nodes in a simple, easy, and transparent way, as well as visualize how each node executes its processes, becomes a very useful and didactic tool.

In sections tree, we describe the implementation of JPeer to represent, configure, and communicate the nodes with different languages programming, as C++ and Java, well as how the distributed algorithms

are used to give access to the resources to each node.

The Framework JPeer distributed nature provide exciting opportunities for new applications to be developed. P2P computing framework proposed distinguishes itself from traditional distributed computing in three main aspects. First, the scalability of P2P systems goes far beyond that of traditional distributed systems. Since P2P systems are able to scale to thousands of nodes, they can harness the power of computers over the Internet. Second, P2P, in its most uncompromising definition, requires everything to be completely decentralized. Ideally, no centralized structures should exist in P2P systems.

Also, the most important, P2P applications often work in highly dynamic environments. Specifically, in terms of network topology, since P2P nodes can join and leave the system anytime, P2P systems do not have a fixed topology. Instead, their topology changes according to nodes in the system. Furthermore, the content and load of system are distributed in real time according to the actual demand and resource capability of nodes.

P2P computing is a important one form of distributed computing. It shares the set of issues that distributed computing researchers have been addressing over the years (e.g., security, trust, anonymity, fault tolerance, scalability, distributed query processing and coordination). However, P2P computing distinguishes itself from traditional distributed computing in several aspects. Some of the most important ones are [8]:

- **Symmetric role.** Each participating node in a P2P system typically acts both as a server and as a client. In fact, each node installs a single package that encompasses both client and server code. As such, a node can issue queries (like a client) and serve requests (like a server).
- **Scalability.** Different from traditional distributed systems, P2P systems can scale to thousands of nodes. As a result, they can harness the power of computers over the Internet. To achieve this property, the P2P protocols do not require all-to-all communication or coordination.
- **Heterogeneity.** A P2P system can be heterogeneous in terms of the hardware capacity of the nodes may be a very slow machine, and another may be a high-end supercomputer.
- **Distributed control.** In its strictest definition, P2P requires everything to be completely decentralized. Ideally, no centralized structures should exist in P2P systems.
- **Dynamism.** P2P applications often work in highly dynamic environments. The topology of P2P systems may change very fast due to joining of new nodes or leaving of existing nodes. The content and load of P2P systems typically change according to the actual demand and resource capability of nodes.

2. STATE OF THE ART

Currently, it has the need for communication distributed applications remotely via videoconferencing, such as distance education and online business. There are some examples based on P2P architecture like a Framework for E-Learning [9]. This paper describes the design and development of a Peer-to-Peer Presentation System that supports live video streaming of lectures coupled with a shareable whiteboard. The participant may resolve doubts during an ongoing session of a presentation using ask doubt feature. It is disseminated to all other peers by preserving the causality between ask doubt and its resolution. A buffered approach was employed for enhancing the performance of the shareable whiteboard which may be used

either in tandem with live streaming or in standalone mode. The proposed P2P-PS supported also by a P2P file sharing and searching system with a few additional features. The combined framework allows creation of mash-up presentations with annotations, comments, posts on video, audio, and PDF files.

In [10], the authors describe the approach to the problem of consensus and agreement: The agreement between processes in a distributed system is a fundamental requirement for a wide range of applications. Many forms of coordination require processes to exchange information, which is used to communication processes with processes and reach a common understanding, before taking specific actions of the application. In [11], it is discussed the application of the results for the problem of consensus based on multi-agent systems, where two algorithms based on the Laplacian matrix of the network G graph are proposed that achieves the consensus in a finite time using Lyapunov functions. In this way, a special distributed map is proposed for the class of non-directed graphs.

These maps are used in [12], where the author propose a comprehensive analysis and design of co-operative strategies for consensus, another contribution is the introduction to the necessary and sufficient conditions for two discontinuous distributed algorithms that achieve minimum and maximum knowledge in finite and asymptotic time. As proposed in this research work, the use of Ad Hoc networks is proposed, for this reason we contemplate these works mentioned above, to validate their results in networks with intercommunication topologies and dynamic changes. Another work with novel approaches to consensus algorithms is [13], in this paper was re-addressed the concomitant problem for distributed non-linear multi-agent systems and apply a controller to directed and non-directed graphs. In addition, this control allows you to work on fixed and changeable topologies. The application of consensuses and agreements in multi-agents in distributed environments and design of observers can be found in [14], this work is based on L neighborhood rules for the coordination of multi-agents. In the search for an active leader, we describe the agent dynamics to be followed with interactive control inputs.

An example of MDS applications can be found in [15], [16], these papers focus on the problem of determining an optimal route, through a route discovery algorithm.

Another similar framework is PeerSim [17]. This is an extremely scalable simulation environment that supports dynamic scenarios such as churn and other failure models. Protocols need to be specifically implemented for the PeerSim Java API, but with a reasonable effort they can be evolved into a real implementation. Testing in specified parameter-spaces is supported as well. This are executed before the simulation, while controls are executed during the simulation. They may modify or monitor every component. For example, they may add new nodes or destroy existing ones; or they may act at the level of protocols providing them with external input or modifying their parameters. Controls can also be used to passively monitor the simulation. PeerSim reads the configuration file and loads the specified classes at run-time. Based on the configuration file, either the cycle-driven or the event-driven simulation engines are loaded. The former, to allow for scalability, uses some simplifying assumptions such as ignoring the details of the transport layer in the communication protocol stack. The latter is less efficient but more realistic. Among other things, it supports transport layer simulation as well. Cycle-based protocols can also be run by the event-based engine, but not upside down.

In [18], the authors developed embedded software libraries in a community cluster for the use of MPI with a well-defined specific

interface available on many platforms and in several programming languages. With the need to use languages oriented to objects, such as the realization of multilanguages. Internally, they performed the JVM Java object translation through the native JNI interface. Once in the native memory space, mpiJava used a native implementation of MPI. Unfortunately, the JNI performs poorly because most of the data transferred between the virtual machine and the native space must be copied. Essentially, the mpiJava implementation is a Java wrapper for a native implementation of MPI. We avoided the JNI overhead in our experiments by tightly integrating mpiJava with a version of Hyperion. This integration also ensured that our implementations of the two Java cluster approaches used a common method for executing Java byte codenamely, Hyperions technique of translating bytecode to C and then C++ to native code. Although knowing that is true using low level languages such as C and C++ in expert areas as high-level parallelism as shown in [19], where they explain it satisfied with directive parallelization tools (based on the OpenMP standard, MPI, PVM, etc.). They explain in their work which are supported by modern compilers, as well as with effective parallel libraries for many application domains. This, however, cannot fully replace high-level means for parallel and concurrent programming at the language and STL levels. Even knowing that there are suitable languages for the area of parallelism, concurrency among others, it's also necessary to adapt languages with native characteristics and take advantage of both advantages.

In [20], they showed difference in performance and uncover the cause. This could provide help to Android application developers for building Android applications efficiently. The Java Native Interface (JNI) is used to call native applications and libraries that are written using C/C++ in Java code. They made a Mibench code native shared library by using NDK-build. Fig. 1 shows the structure of the application execution by using JNI of Android NDK. The result of this experiment shows that Android applications that use the native shared library through the JNI of the Android NDK is faster compared to those applications which are compiled by using the native cross-compiler in five of six cases. It seems that the difference in libraries used also makes a difference in the performance. Android NDK uses the bionic C/C++ library, but the native cross-compiler uses the GNU C library. The bionic C/C++ library is developed by Google for Android embedded systems. It is generally small and fast as it is optimized for embedded environments.

In [21], they tested 5 different methods: JNI Communication Delay, Integer Calculation, Floating-Point Calculation, a memory access algorithm, and a heap memory allocation algorithm. As a result, Lee [22] also presented that the Android applications that use the native shared library are faster compared to the Android applications using only Java language in the Android virtual machine. They experimented with 14 different algorithms on a real Android mobile phone [23]. According to the results, the overall performance of an Android application using the JNI of Android is faster than the Dalvik Java code.

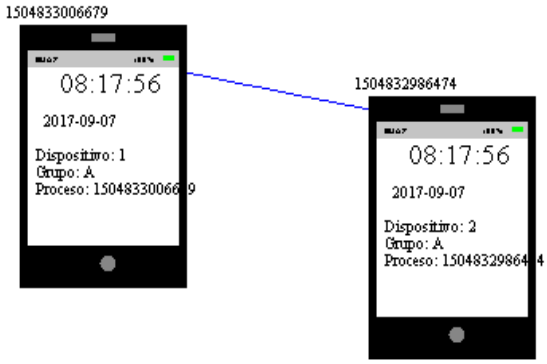


Fig. 1. Representation of two nodes with a single connection.

Programming methodology in [24] is used at native level using the implementation of an interface allowing to represent that any mobile distributed system was achieved.

Nevertheless, a visual environment that facilitates this tracking would be very helpful. This work describes the design and implementation of a didactic graphical interface for a distributed embedded software, which lets the visual representation of a process scheduling algorithm in a virtual mobile distributed system. So far, just one process can be executed by each node, but it is planned to modify the functions to allow the generation of more processes. Also, the use of other process scheduling algorithms is considered.

3. ANALYSIS FOR THE RESOURCE AND SERVICES

Embedded systems [25] have traditionally been differentiated from desktop systems on the basis of functionality. Another consideration is the size of the complete Java platform. Many people believe that support of a Java VM is all you need to run Java applications. However, to compute the total size of the Java platform correctly, you must add the size of the Java VM, Java API package, Java application, and associated native-code libraries. Current Java API packages tend to be large, so the specific API selected for your device will significantly impact its size. Added components can also affect platform size. The functionalities identified were: a) the node network creation (being the nodes the mobile devices), and b) the obtaining of the required information to implement a process scheduler under a virtual environment. The following subsections describe how these functionalities were achieved.

In [1], [2], to represent a virtual mobile device some considerations were taken into account and to insert as many mobile devices as possible. The first was achieved by using a simple rectangle to represent a node, straight lines for its connections (see Fig. 1) and the use of the right click of the mouse for editing options. It is important to mention that on the back-end of the software, a single adjacency list with four elements (transmitter node, receiver node, bridge node and router node), was used to store the connections. The latter was accomplished combining the date and time and generating a unique identifier for each node.

In an MDS, the nodes can communicate with neighbors by means of a suitable configuration, which allows us a multicast, in a transmission range from transmitter to receiver within a neighborhood L, established in [27]. We use the formalism in [16], communication in this MDS configuration neighborhood describes a dynamic topology, which makes it difficult to establish and maintain a communication path between the nodes $v_i \rightarrow v_j$, taking into account that $i \neq j$ and v_i, $v_j \in V$, where V is the set of nodes or peers in a mobile network, presumably remote because of the unwillingness to communicate in a given time.

Assuming that the solution of using a related cyclic graph, reduces these errors in planning and communication among peers, Eq. (1), as was proposed in [16].

$$G=\langle P, R, J, A\rangle, p_i \in P, r_k \in R \text{ and } j_l \in J \qquad (1)$$

where:

G: is a connected acyclic graph.
A: is the edge between peers.
P: processes.
R: resources.
J: task.

It should be noted that 92% of the peers achieved their deadline established a priori. From 980 peers, a constant with respect to time is observed. The interpretation of this statement indicates that the scheduler has a brief improvement with respect to the deadline of the peer´s processes. After a processing time, 98.9% of processes reaching out their deadline are achieved. The remaining 8% achieved a considerable improvement in the range of 1.8s to 2 seconds, and it keeps up with this behavior.

Finally, improvements are expected to this proposed algorithm increasing the number of processes and resources. In an MDS, there are nodes as a dynamic system, therefore, it is necessary to perform the analysis of the metric to be used, which is the minimization of the sum of the weighted weight in determined times, based on a cost of send a packet between two nodes, as expressed in equation (2). This metric is important since in the MDS the difference of values is applied to the real-time systems as shown in [5], [8]. A metric is also proposed, to compare different algorithms based on theorems of process scheduling on uni-processors and multiprocessors, by measuring the computation time required for the determination of a planner that satisfies the partial order and resource constraints.

$$M_c = \frac{1}{n}\sum_{i=1}^{n-1} N_{p_i}(t_c) - N_{p_{i+1}}(t_c) \quad (2)$$

$$t_c = max(f_i) - min(a_i) \quad (3)$$

where:

M_c: proposed metric.
$N_{pi}(t_c)$: node weighting i.
t_c: total completion time.
f_i: end time of tasks.
a_i: start time of the task.

Observe that the total completion time as defined in Eq. (3) considers the time of the task that took longer to finish and the time of the task that started first. Expanding the metric expression, many of the terms cancel out in the telescopic sum (4), and we have:

$$M_c = \frac{1}{n}(N_{p_1}(t_c) - N_{p_2}(t_c) + N_{p_2}(t_c) - N_{p_3}(t_c) + \ldots + N_{p_{n-1}}(t_c) - N_{p_n}(t_c)) \quad (4)$$

The select analysis of task scheduling metric It is taken as a fundamental basis how the development of the core of the system. This takes part of a metric of total time of tasks completion and the metric based on time sum weighted. They are important of the individual characteristics of time for each deadline and by taking the relevance of having different types of tasks taught in the proposed system.

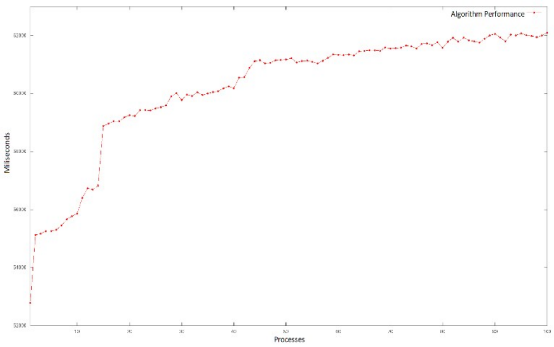
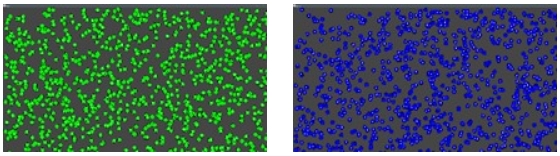


Fig. 2. Initial performance of the algorithm with 100.

Fig. 2 shows the start of the executions of the tasks from 1 to 100 processes. The first processes enter the deadline in good conditions.

The executions were performed on a node assigned by the LNS-BUAP supercomputer lab [28], using instances of the reduced operating system. The kernel has basic resources of an operating system, being ideal for the execution of the algorithms of planning. These instances were carried out with a tool for the administration of the cloud of computation, several tests were performed with at least 1,000 instances of peers and a handling of 1,000 processes.



(a) (b)

Fig. 3. (a) Peers in Java. (b) Peers in C++.

In Fig. 3, we show the P2P network. In a) there are more than 100 peers in Java under the JVM. In b) there are peers in C ++ with graphical environment of Java, which implies the use of JNI, so the interface between the

language at low level with the interface of the graphic environment (Listing 1.1).

Listing 1.1. RTPeerinC.jni

```
#include <jni.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
JNIEXPORT void JNICALL Java_DelayTime
   (JNIEnv * env, jobject jobj,jstring i){
   int ID;
   time_t rawtime;
   struct tm *timeinfo;
   time (&rawtime);
   timeinfo=localtime(&rawtime);
   const char *str=(*env)
      ->GetStringUTFChars(env, i, 0);
   printf("Dispositivo:\%d Group:
      Processes:\%s Date:\%s\n",ID++,str,
      asctime(timeinfo));
}
```

In Fig. 4, we can see the saturation of peers under tow different programming languages for the evaluation of frameworks with respect to the limits of resources and communication services through messages. The peers in green are communicated with the peers in blue with the help of JNI. This native interface was very useful to obtain the real data that the peers throw in C ++ and be able to communicate with the peers under Java, as well as being visible in a user graphical environment with the JVM.

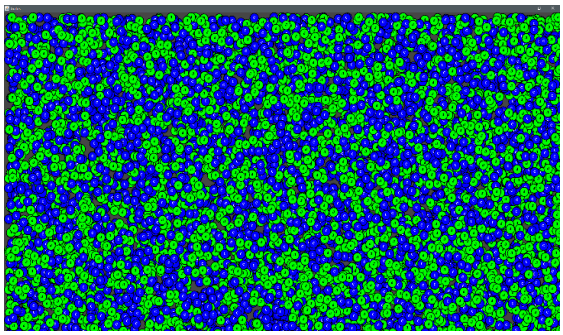


Fig. 4. Communication between peers in C ++ and Java.

4. RESULT

For the creation and manipulation of each peer on the JPeer, a contextual control menu can be used to display a set of possible actions: a) New Peer C++; b) New Peer Java; c) Individual elimination; d) Overall elimination; and e) Inter-communication JNI. Fig. 3 shows Java Native Interface (JNI) function, that allows the native calls for the communication between the embedded software and the underneath operative system.

The main functionality of JPeer Framework is the communication between the virtual mobile devices through an information flow between two layers, the graphic layer and the low-level layer. For this reason, a multiplatform with JNI and the supercomputing was implemented. In JPeer, the information exchanged between the nodes includes the identifier, the execution time (start and final) and the quantum assigned by the system to the process. The code shown in ((Listing 1.2)) correspond to the function that requests for time in the java native interface and assign the respective peer to the node.

Listing 1.2. PeerJavaandC.jni

```
#include<jni.h>
#include<stdio.h>
#include "MainWindow.h"
#include<String.h>
//Implementation of native method
//JNI class
JNIEXPORT jstring JNICALL
        Java_MainWindow_PeerInC
        (JNIEnv *env,jobject thisObj){
  char msg[60]="C++";
  jstring result=(*)NewStringUTF(msg);
        return result;
}
```

For representing the JPeer Framework, a driver capable of using real resources was implemented as a DLL file. When the JNI option is selected in the contextual menu, the driver requests the ID processes of the connected nodes from the supercomputer working on real time. Then a command line terminal is used to verify that the exchange of messages between the connected nodes was correct. Because a personal computer did not have enough resources for testing the creation a lot of mobile devices, there was the need to use a supercomputer, whose characteristics are described next: The Cuetlaxcoapan supercomputer of the LNS [11], is composed of a standard calculation cluster with Intel Xeon processors and a cluster with Intel Xeon Phi Knights Landing

processors with 228 Thin calculation nodes (5472 total cores). Each node contains 2 Intel Xeon E5-2680 v3 processors (Haswell) at 2.5 GHz, 12 cores per processor / 24 total cores, 128 GB of DDR4 memory at 2133 MHz, 2 Gigabit Ethernet network interfaces and an InfiniBand FDR 56 Gbps net-work interface. Storage of 1.2 PB of total space for disk storage. Finally, there is a Gigabit Ethernet network for managing the hardware of the supercomputer and the provisioning of software to the nodes.

The implementation of a user interface that allows to represent any paradigm of distributed system was achieved. The networks P2P on between them, and the information in-side each node correspond to the resources requested to the system. The interface is centered in graphs edges and relations between the networks.

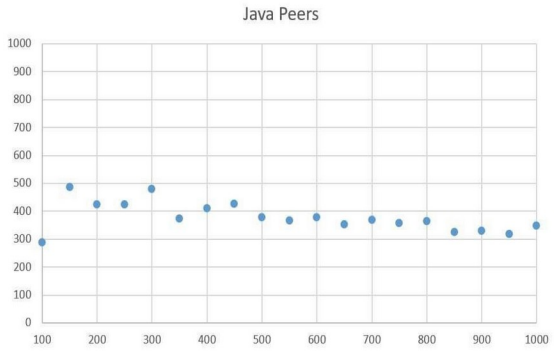


Fig. 5. The balance of the load of the cluster.

In Fig. 5, we show the balance of the load of the cluster in the time the use of the same time while the number of companions grows with 100 pairs the use of the computer of 288 per second, while with 10000 it is of 347 of 6 to 7 seconds. The start and launch time are approximately 13.4 seconds.

5. CONCLUSION AND DISCUSSION

The graphical user interface implemented in this project allows us to visually represent one or more P2P networks in a mobile distributed system, to show an easy and transparent interaction between various nodes. A metric was used to measure the communication time between peers in one neighborhood L. The metric is justified by the response time in its deadlines of each process for low-level communication with the native interface JNI. For this reason, it is possible to visualize how a multi-languages P2P network is executed through the exchange of information between the nodes by means of labels inserted in each node.

Until now, each node can execute a single process, but it is planned to modify the functions to allow the generation of more peers. Knowing that there are adequate languages for the area of parallelism, concurrency among others, it is also necessary to adapt languages with native characteristics and take advantage of their important characteristics. This project proposes a framework for educational use, as well as scientific and technological use.

ACKNOWLEDGMENT

The authors thankfully acknowledge computer resources, technical advice and support provided by National Supercomputing Laboratory of Southeast Mexico (LNS), a member of the CONACYT national laboratories, with project No. 201901014C. Also too, Laboratory of Technology and innovation of FCC-BUAP..

REFERENCES

[1] Scott Corson, M., J. Macker, J. and Batsell, S. G. Architectural considerations for mobile mesh networking. *Military Communications Conference, 1996. MILCOM'96, Conference Proceedings*, IEEE. Vol. 1. IEEE, 1996. https://doi.org/10.1109/MILCOM.1996.568618

[2] Park, V.D. and Scott Corson, M. A highly adaptive distributed routing algorithm for mobile wireless networks. INFOCOM'97. *Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE*. Vol. 3. IEEE, 1997. https://doi.org/10.1109/INFCOM.1997.631180

[3] Dinh, H.T., et al. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing* 13.18 (2013): 1587-1611. https://doi.org/10.1002/wcm.1203

[4] Stankovic, J.A., et al. Implications of classical scheduling results for real-time systems. *Computer* 28.6 (1995): 16-25. https://doi.org/10.1109/2.386982

[5] Bini, E. and Buttazzo, G. C. Measuring the performance of schedulability tests. *Real-Time Systems* 30.1-2 (2005): 129-154. https://doi.org/10.1007/s11241-005-0507-9

[6] Cigdem, S. and Kravets, R. Bypass routing: An on-deman local recovery protocol for Ad Hoc netwoks. *Ad Hoc Networks* 4 (3), may, 2006 pp. 380-397 Elsevier Science Publishers B. V. Amsterdam, The Netherlands. https://doi.org/10.1016/j.adhoc.2004.10.004

[7] Ramasubramanian, V., Haas, Z.J. and Sirer, E.G. SHARP: A hybrid adaptive routing protocol for mobile ad hoc networks. *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking computing*. ACM, 2003. https://doi.org/10.1145/778415.778450

[8] Vu, Q.H., Lupu, M. and Ooi, B. Ch. *Peer-to-peer computing: Principles and applications*. Springer Science Business Media, 2009. https://doi.org/10.1007/978-3-642-03514-2

[9] Hatcher, P. et al. Cluster computting with Java. *Computing in science engineering* 7.2 (2005): 34-39. https://doi.org/10.1109/MCSE.2005.28

[10] Bhagatkar, N., Dolas. K., and Ghosh, R.K. An Integrated P2P Framework for E-Learning. *arXiv preprint* arXiv:1903.05474 (2019). https://doi.org/10.1007/s12083-020-00919-0

[11] Cheng, A.M.K. *Real-time systems: scheduling, analysis, and verification*. John Wiley & Sons, 2003.

[12] Hong, Y., Chen, G. and Bushnell, L. Distributed observers design for leader-following control of multi-agent networks. *Automatica* 44.3 (2008): 846-850. https://doi.org/10.1016/j.automatica.2007.07.004

[13] Esquivel-Flores, O., Benitez-Pérez, H. and Ortega-Arjona, J. Issues on communication network control system based upon scheduling strategy using numerical simulations. *Numerical Simulation-From Theory to Industry*. IntechOpen, 2012. https://doi.org/10.5772/48578

[14] Cortéz J. Distributed algorithms for reaching consensus on general functions. *Automatica* 44.3 (2008): 726-737. https://doi.org/10.1016/j.automatica.2007.07.022

[15] Esquivel-Flores, O., and Benítez-Pérez, H. Dynamic Reconfiguration of Real-Time Distributed Systems Based on Agents. *Revista Iberoamericana de Automca e Informca Industrial RIAI*. Vol.9, Issue 3, July-September 2012, pp. 300-313.

[16] Castillo, O. and Benítez-Pérez, H. A Novel Technique to Enlarge the Maximum Allowable Delay Bound in Sampled-Data Systems, *Congreso Nacional de Control Automco* 2017 Monterrey, Nuevo León, Octubre 4-6, 2017.

[17] Larios-Gómez, M., et al. A Scheduling Algorithm for a Platform in Real Time. *International Conference on Supercomputing in Mexico*. Springer, Cham, 2018. https://doi.org/10.1007/978-3-030-10448-1_1

[18] Montresor, A. and Jelasity, M. PeerSim: A scalable P2P simulator. *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*. IEEE, 2009. https://doi.org/10.1109/P2P.2009.5284506

[19] Vyukova, N. I., Galatenko, V.A. and Samborskii, S.V. Support for Parallel and Concurrent Programming in C++. *Programming and Computer Software* 44.1 (2018): 35-42. https://doi.org/10.1134/S0361768818010073

[20] Kim, Y.-J., et al. Benchmarking Java application using JNI and native C application on Android. *2012 12th International Conference on Control, Automation and Systems*. IEEE, 2012.

[21] Lee, S. and Jeon, J. Evaluating Performance of Android Platform Using Native C for Embedded Systems, *2010 International Conference on Control Automation and Systems*, pp. 1160-1163, 2010. https://doi.org/10.1109/ICCAS.2010.5669738

[22] Lee, J.K., and Lee, J.Y. Android programming techniques for improving performance. *2011 3rd International Conference on Awareness Science and Technology (iCAST)*. IEEE, 2011. https://doi.org/10.1109/ICAwST.2011.6163105

[23] Lin, C.M., Lin, J.H., Dow, C.R. and Wen, C.M. Benchmark Dalvik and Native Code for Android System, *2011 Second International Conference on Innovations in Bio-inspired Computing and Application*, pp. 320-323, 2011. https://doi.org/10.1109/IBICA.2011.85

[24] Larios Gómez, M., Beristain Hernández, A. et al. JScheduling: A Graphical Interface for Applying a Process Scheduling Algorithm. *Research in Computing Science* 145 (2017): 119-125.

[25] Mulchandani, D. Java for embedded systems. *IEEE Internet Computing* 3 (1998): 30-39. https://doi.org/10.1109/4236.683797

[26] Vlachou, A. et al. Peer-to-peer query processing over multidimensional data. *Springer Science Business Media*, 2012. https://doi.org/10.1007/978-1-4614-2110-8

[27] Vu, Q.H., Lupu, M. and Ooi, B. Ch. Architecture of peer-to-peer systems. *Peer-to-Peer Computing. Springer*, Berlin, Heidelberg, 2010. 11-37. https://doi.org/10.1007/978-3-642-03514-2_2

[28] LNS-BUAP. Homepage, http://lns.org.mx/?q=content/proyectos-aceptados

ACERCA DE LOS AUTORES

Mariano Larios Gómez. Originario de Puebla-México. Profesor investigador tiempo completo en la Benemérita Universidad Autónoma de Puebla (BUAP) con perfil PRODEP. Recibió su grado de licenciatura y maestría en ciencias de la computación en la facultad de ciencias de la computación (BUAP) 1997-2001 y 2001-2003 respectivamente. Estudios de doctorado en sistemas en Sistemas Inteligentes en la UATX 2019. Desde 2004 es profesor en la facultad de ciencias de la computación (BUAP). Su interés en la investigación incluye tópicos en cómputo distribuido, blockchain, cómputo de alto rendimiento, sistemas de tiempo real y computo pervasivos. Actualmente colabora en proyectos de investigación sobre supercómputo en el laboratorio nacional del suroeste LNS.

Adriana Hernández Beristain. Realizó sus estudios de Licenciatura en el IT de Tehuacán (1994-1999) en la carrera de Ing. en Sistemas Computacionales, posteriormente realizó sus estudios de maestría en la Facultad de Ciencias de la Computación de la Benemérita Universidad Autónoma (2001-2004) de Puebla. Sus intereses sobre la investigación son sobre los sistemas de Información y Comunicación, Sistemas Distribuidos; Seguridad de redes y VoIp. Ha participado en varios proyectos de investigación, entre los que destaca: "Entorno para la comunicación efectiva por Telepresencia entre las dependencias de H. Ayuntamiento de Puebla" en donde Desarrollo un entorno de comunicación utilizando la innovación tecnológica y la TelePresencia basada en CISCO. Actualmente labora como Profesor Investigador TC en la BUAP, imparte cursos en el área de las redes de computadoras y de certificación en CCNA de CISCO, es coordinadora de la academia Cisco, ha publicado varios artículos en revistas y memorias en extenso.

Mario Anzures García. Profesor Investigador Titular en la Facultad de Ciencias de la Computación (FCC) de la Benemérita Universidad Autónoma de Puebla, México desde 1995. Obtuvo el grado de Maestría y Doctorado en Tecnologías de la Información y la Comunicación en la Universidad de Granada, España. Es miembro del Sistema Nacional de Investigadores (SNI) en el área 1 de Ciencias Físico Matemáticas y Ciencias Materiales del CONACYT, del Padrón de Investigadores VIEP-BUAP y del PRODEP. Ha sido coordinador de la Ingeniería en Tecnologías de la Información de la FCC desde 2012 hasta 2019. Él ha publicado más de 40 artículos en conferencias y revistas nacionales e internacionales en las líneas de investigación: Modelado Arquitectónico y/o Ontológico para el Desarrollo de Groupware, Metodologías de Desarrollo de Software basada en Patrones de Diseño y Tecnologías de la Web Semántica e Inteligencia Artificial. También es miembro de la Red Latinoamericana en Tecnologías Concurrentes, Distribuidas y Paralelas; y de la Sociedad Mexicana de Inteligencia Artificial (SMIA). Así como del Comité del Programa de diferentes conferencias y revisor de varias revistas nacionales e internacionales.

Actualmente, está trabajando en varios proyectos de investigación.

Franco Rojas López. Profesor de tiempo completo en la universidad Politécnica Metropolitana de Puebla. Recibió el grado de maestro en ciencias de la computación en la Benemérita Universidad Autónoma de Puebla, posteriormente el grado de doctor en el Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional. Sus áreas de interés son representación y manejo de conocimiento, sistemas de recomendación y chatbots.

Erica A. Martínez Mirón. Actualmente Profesora Investigadora de Tiempo Completo en la Facultad de Ciencias de la Computación de la Benemérita Universidad Autónoma de Puebla. Realizó sus estudios de Licenciatura en la Benemérita Universidad Autónoma de Puebla (MX). Obtuvo su maestría en el Instituto de Investigación en Matemáticas Aplicadas y Sistemas de la UNAM (MX) y el grado de Doctora en la Universidad de Sussex (GB) en el área de Ciencias de la Computación. Sus líneas de investigación versan sobre la Interacción Humano-Computadora; en particular sobre el trabajo colaborativo, el cómputo afectivo, así como la definición de modelos de usuario y su proceso de aprendizaje y evaluación en ambientes interactivos educativos. Ha participado en varios proyectos de investigación, impartido diversos cursos curriculares, de actualización y preparación a nivel de licenciatura, maestría y doctorado y ha publicado varios artículos en revistas indexadas y memorias en extenso en el área de inteligencia artificial y educación.