www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 15(5), Dec 2024, 84-95. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2024.v15i5.561

A Strategic Approach to Metric Management for a Real-time Process Based on a Task Scheduling Algorithm in Distributed Mobile Environments

Mariano Larios Gómez [1], *Perfecto Malaquías Quintero Flores* [2], *Mario Anzures García* [1]

[1] Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación, México.
[2] Universidad Autónoma de Tlaxcala, Facultad de Ciencias Básicas, Ingeniería y Tecnología, Posgrado en Ciencias en Sistemas Computacionales y Electrónicos, México.
mariano.larios@correo.buap.mx

Abstract. This project introduces a metric-driven strategy for enhancing real-time task planning within a mobile distributed system. The core design revolves around a real-time task planning algorithm that exploits consensus mechanisms among multiple nodes. By incorporating innovative real-time scheduling algorithms, validated on a supercomputer, the project establishes frameworks for simulating distributed mobile environments in real-time. This enables the execution of tasks in unpredictable contexts, ensuring adherence to stringent time constraints and empowering decision-making capabilities. The primary objective is to optimize task allocation in mobile distributed systems, including aerial nodes, thereby facilitating seamless data transfer while maintaining data integrity. The project builds upon the Fan task scheduler, integrating compensatory measures for online load distribution and optimization of message routes, effectively reducing communication latency in dynamic mobile environments. The contributions made aim to enhance real-time task allocation across diverse nodes, enabling the transfer of location and search data across networks without any data loss. This approach bears significant importance for the efficient management of tasks within dynamic scenarios.

Keywords: Real-time Systems, Distributed Environments, Embedded System, Scheduling Tasks.

Article Info
Received May 10, 2024.
Accepted Nov 20, 2024.

1 Introduction

Real-time systems play a critical role in modern technologies, particularly in the context of dynamic mobile distributed networks. The scientific community recognizes the importance of real-time distributed activities, emphasizing optimization aspects such as processing, memory, communication, energy, timing, and precision, as discussed in [1, 2]. These sources also highlight how real-time systems provide valuable feedback mechanisms for users while addressing the challenges posed by real-time operations in distributed environments, especially concerning data transmission.

Efficient recognition systems in such environments rely heavily on robust architectures due to the disruptions common in congested mobile distributed environments with limited resources. A novel solution has emerged to address these challenges by introducing a router to oversee and establish the Fan task scheduler. This

© Editorial Académica Dragón Azteca S. de R.L. de C.V. (EDITADA.ORG), All rights reserved.

scheduler integrates online load distribution compensation, message route optimization, and communication time reduction through routing enhancements [3-5, 20].

In a mobile distributed system (MDS), nodes form a dynamic system, requiring the selection of appropriate metrics. The objective is to minimize the sum of weighted weights at specific times, considering packet transmission costs between nodes, as expressed in equation (3). This metric plays a crucial role within the Weighted Mobile Distribution (WMD) context, where value differences are integrated into the complementary system [6, 7]. Building upon previous work [8-10], we propose a metric to compare algorithms based on process scheduling theorems for uniprocessors and multiprocessors, measuring the computation time required for scheduler determination under partial order and resource constraints.

Entities within an MDS include sending, receiving, and bridging/router nodes, adapting based on communication needs to form local states. Nodes communicate within a dynamic topology, employing message flooding and route search mechanisms within a designated neighborhood (L), utilizing Ad Hoc Sustainable Longevity Routing (SLR) networks [9, 10].

The analysis, design, and implementation of a planning algorithm ensure the execution of periodic, aperiodic, and sporadic tasks within simulated dynamic environments, adhering to deadline constraints and implementing the planned Earliest Deadline First (EDF) approach [11, 12]. Extensive literature reviews on mobile distributed systems, real-time systems, and high-performance computing, combined with the application of hybrid routing and reactive protocol algorithms, led to the application of algorithms on a real-time execution-capable mobile platform.

Simulated algorithms and methodologies were developed within a high-performance computing architecture, utilizing a supercomputer with ample resources for tests and desired outcomes. Within the MDS, the configurations of sender, receiver, and bridge-router nodes were investigated. These nodes interact within neighborhoods (L) through adaptable configurations, allowing multicasting within transmission ranges, as discussed in [16, 17].

Ultimately, the objectives of the project include developing algorithms for real-time task planning within aerial mobile distributed systems, benefiting drone communication, and enhancing responses in human-risk scenarios such as natural disasters. The article proposes embedded tools for system analysis and real-time planning algorithm results in distributed computing environments, supported by an embedded software tool that measures real-time task and process planning times. This tool is tested under risky scenarios, effectively making decisions based on real-time planning tests [18-25].

2 State of the Art

Communication networks involving mobile devices and aerial robotics, such as drones, are pivotal across various applications, including goods delivery, surveillance, search and rescue, and even entertainment. Acknowledging the potential utilization of drones in future wars and regional conflicts [29], the advancement of drone-to-drone (D2D) networks holds promise for enhancing the efficiency of these applications.

One approach to bolster D2D network performance focuses on monitoring the channel utilization between drones, as highlighted in [30]. Within the context of smart cities, where wireless network characteristics and drone mobility present communication challenges, a protocol is proposed to address the hidden terminal issue. This protocol operates through search and functional sublayers.

In [31], the authors advocate a consensus and agreement approach, underscoring its importance in distributed systems for various applications. Effective coordination often hinges on information exchange between processes to establish shared understanding before specific actions are taken. Real-time planning in mobile devices finds applications in healthcare and everyday scenarios. An instance is the application of communication and process planning in drones for autonomous pigeon deterrence [26]. This approach aims to minimize damage to structures and mitigate the spread of pigeon-borne diseases. Data collected from these efforts can aid in comprehending the intricate interplay between pigeons and drones.

Airborne mobile communication also plays a crucial role in agriculture, addressing labor shortages and agricultural losses due to climate change and pests. Precision agriculture technologies are undergoing transformation through the integration of electronics and mechanics, capitalizing on Unmanned Aerial Vehicles (UAVs) [32]. A multi-rotor drone system is proposed to address the challenges faced by farmers in eliminating potential threats. This study introduces an affordable and lightweight drone system suitable for low-altitude flights over crop fields.

Hence, the proposal to develop real-time planning algorithms holds immense significance. It enables remote communication, efficient operation, and effective coordination among devices and mobile robots. This is particularly pertinent in urban airspace, where D2D interactions present distinct challenges [33]. Addressing conflicts and traffic density in drone traffic can benefit from the utilization of Machine Learning (ML) techniques, including deep learning via multilayer neural networks. This research experimentally manipulates traffic load, predictability, and lead time to potentially predict and mitigate conflicts.

The explosive growth of mobile devices and the rapid development of wireless networks and mobile computing technologies have catalyzed the emergence of diverse computing paradigms, including Fog Computing and Mobile Cloud Computing (MCC), aiming to enhance mobile applications' Quality of Service (QoS) by enabling computation offloading to the edge cloud and leveraging idle computation capabilities of mobile devices. Efficiently scheduling offloaded tasks becomes imperative, particularly in scenarios with limited computation, storage, communication resources, and energy supply. In this context, the Cooperative Multi-tasks Scheduling based on Ant Colony Optimization algorithm (CMSACO) presented in this study addresses the multi-tasks scheduling problem within hybrid MCC architecture. Through optimization formulation and consideration of factors such as task profit, deadline, dependence, node heterogeneity, and load balancing, CMSACO demonstrates superior efficiency over existing algorithms. The study underscores the significance of developing efficient scheduling algorithms in MCC and Fog systems to attain objectives like increased profit and reduced energy consumption, focusing specifically on multi-task scheduling with time constraints and proposing a heuristic ACO-based solution. Despite its static nature catering to batch scheduling, future work will extend the algorithm to address online scheduling challenges [39, 40].

In [41] Cloud computing is rapidly advancing, necessitating efficient resource allocation and equitable task distribution to achieve optimal performance and cost-effectiveness. In this context, EcoSched emerges as a groundbreaking initiative within cloud computing, aiming to redefine resource allocation and task scheduling methodologies for improved efficiency and sustainability. This study delves into dynamic task scheduling methods tailored to optimize resource utilization and task distribution in cloud environments, catering to evolving demands. The innovative framework of EcoSched prioritizes eco-efficient task assignment by categorizing tasks based on computational intensity, interdependencies, and strict deadlines. Leveraging a refined task assignment mechanism and a sophisticated dynamic task scheduler, tasks are intelligently allocated to suitable virtual machines in real-time. Additionally, heuristic and predictive analyses augment decision-making within the scheduler, ensuring optimal task placement. Simultaneously, EcoSched integrates a robust load balancer capable of dynamically adjusting task allocations across the cloud infrastructure, preemptively mitigating resource bottlenecks and minimizing response times to significantly enhance system performance. Notably, the proposed methodology demonstrates substantial improvements in response time and resource utilization metrics, outperforming conventional scheduling approaches. This research offers invaluable insights into the scalability and adaptability of the introduced techniques, laying the foundation for future advancements in dynamic task scheduling strategies. By prioritizing resource allocation optimization and load balancing, this study contributes to the development of resilient, efficient, and sustainable cloud environments. EcoSched marks a significant milestone in addressing escalating computational demands while championing eco-efficiency, thus shaping the future landscape of cloud computing.

In [42, 43] the quest to enhance resource availability and minimize the total execution time and energy consumption in cloud task scheduling, the utilization of electric fish optimization (EAEFA) emerges as a promising solution. By leveraging the inherent characteristics of electric fish, which exhibit attraction towards high-quality solutions and aversion towards low-quality ones, EAEFA effectively addresses the multi-objective optimization problem inherent in task scheduling. Through extensive experimentation on real-world workloads, including simulations on HPC2N, EAEFA showcases remarkable improvements, surpassing state-of-the-art methods by over 30% in makespan time and more than 20% in overall energy consumption. This underscores the efficacy of EAEFA in minimizing makespan, energy consumption, and resource utilization while maximizing load balance. The success of EAEFA in tackling the challenges of cloud computing work scheduling suggests its viability as a solution for the broader cloud computing landscape. Looking ahead, further research can explore the integration of EAEFA into other aspects of cloud computing, such as resource allocation, management, network routing, and load balancing, with potential enhancements through hybrid techniques combining EAEFA with complementary optimization algorithms.

3 Metric Design and Application

The Fan algorithm, employed in implementing the metric (refer to Fig. 1), accentuates the control, generation, and progression of tasks within designated timeframes. These tasks encompass message transmission at specific instances, establishing a process for each message, thus culminating in the metric's derivation concerning time. Each process was distinguished by an identifier; when the value diverges from zero, it extends the initiation of subordinate processes, thereby enhancing task arrangement through the Fan algorithm. Conversely, in the absence of such conditions, the processes remain suspended.

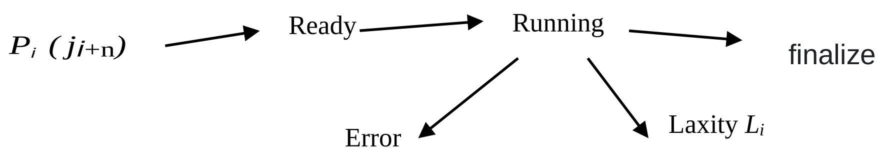

Figure 1. Calling of processes with a specific deadline is a key aspect of this study.

In the planning of each process, they are recognized as ready processes with a set of Jn tasks. Subsequently, the creation of P_m processes is expanded, ensuring the avoidance of entering the error state (E) if the deadline is adhered to. In equation (1), the following condition is presented: if the slack time (lost time) X_j of each process exceeds the sum of the absolute constrained times d_i, then the process will be put on hold.

$$E = \sum_{i=1}^{n} (d_i) \leq X_j \tag{1}$$

where:
X_j: the time by which a task can be delayed in its activations to meet its deadline.
d_i: absolute deadline.

The goal is to circumvent the planning issue known as the domino effect, by introducing a real-time scheduler σ(t) in which processes are generated following a spanning tree structure. This involves sets of processes P, resources R, and tasks J, and A the connection between processes, resources, and tasks, as a connected acyclic graph G < P, R, J, A >.

Utilizing a connected acyclic graph as part of the solution helps mitigate these planning errors, as suggested in [36, 29], which serves as the primary reference for this article.

The experiments were conducted on a node provided by the LNS-BUAP (by its acronym in Spanish, Laboratorio Nacional del Sureste - Benemérita Universidad Autónoma de Puebla) supercomputing laboratory, utilizing instances of the lightweight operating system with a micro-kernel known as Minix. The micro-kernel offers essential operating system resources, rendering it ideal for executing scheduling algorithms. These instances were generated using a cloud computing administration tool called OpenStack. Through OpenStack, a substantial number of Minix instances were executed, encompassing numerous tests involving at least 1,000 instances, and the management of 1,000 processes.

Based on the operation of the Fan algorithm, a deployment initiates a thousand threads executing tasks for each created process within a range of eight hundred processes, alongside the application of the Lamport algorithm, as discussed in [34, 35]. Initially, the thread's rotation count is calculated, which is determined by the highest number of rotations among other threads, plus one. Before entering its critical section, the thread ensures obtaining the lowest number.

Because of implementing the Lamport algorithm, a revised version of the original Fan algorithm code was derived, ensuring successful completion for a significant portion of the created processes and their associated threads, along with their execution times. The execution time of tasks increases in segments of one hundred processes, evident in the graphs Fig. 2 and Fig. 3. Notably, the segment spanning from P_{701} to P_{800} exhibits the longest execution times due to the planning executed by the Lamport algorithm.

4 Results

The algorithm proposed here operates on the fundamental principle of initializing all processes with predefined deadlines, organizing them akin to a fan to distribute them effectively and ensure the accomplishment of their respective objectives.

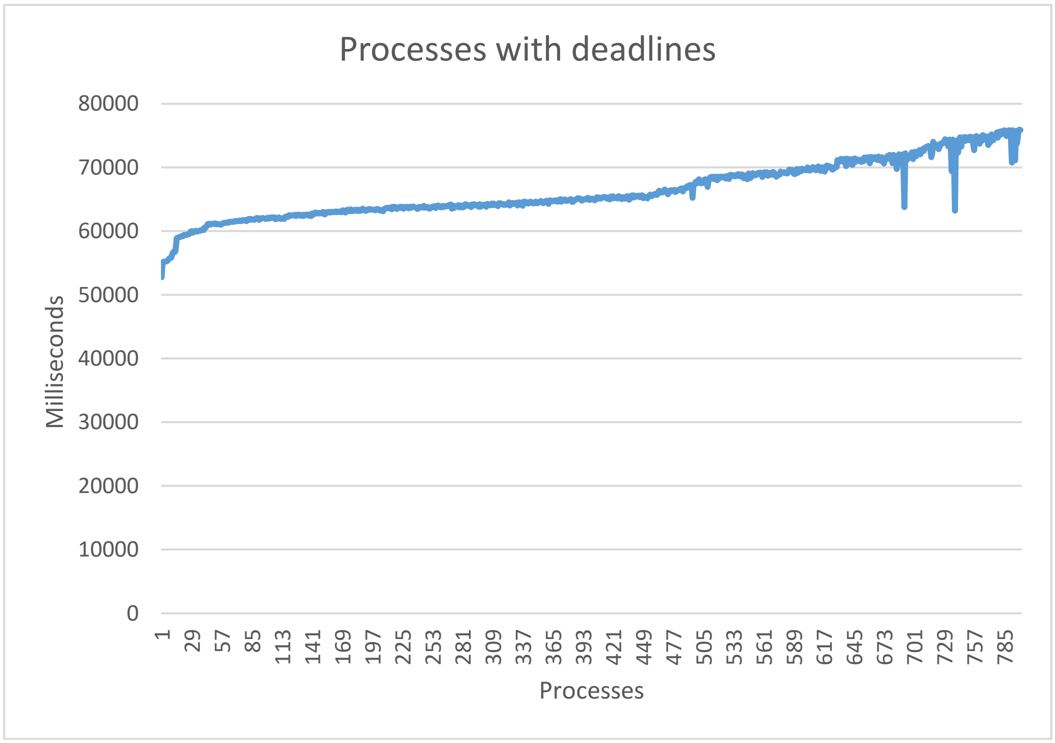

Figure 2. Provides a visual representation of the assignment of priority levels to processes based on their time constraints and criticality.

During the conducted tests, we meticulously measured the time required for all processes to meet their deadlines. As depicted in Figure 2, an impressive 92% of the processes successfully met their predefined deadlines within one minute, while the remaining processes concluded within two hours.

To further validate and refine this performance, additional testing is necessary. This includes exploring the impact of scaling up the number of processes and allocating additional resources. Nevertheless, the initial results obtained from the proposed algorithm offer valuable insights and can serve as robust benchmarks for comparative analyses against similar algorithms. These comparisons will help ascertain whether the proposed algorithm effectively reduces process programming time.

As illustrated in Figure 2, processes are represented in seconds, providing a clear visualization of the outcomes obtained through testing at the LNS laboratory node. Notably, 92% of the processes adhered to the predefined deadlines. Among the pool of 980 processes, a consistent pattern emerged over time, indicating a marginal enhancement in meeting process deadlines facilitated by the scheduler. Following an initial processing duration, an impressive 98.9% of the processes met their deadlines, with the remaining 8% showing notable improvement ranging from 1.8 seconds to 2 seconds. This consistent pattern underscores the algorithm's reliability and effectiveness.

Moving forward, further enhancements to the proposed algorithm will involve expanding the number of processes and resources. This iterative process aims to optimize performance and accommodate dynamic changes in the system environment.

In the context of a Mobile Distributed System (MDS), the presence of dynamic nodes necessitates a comprehensive analysis of the designated metric. This metric entails minimizing the weighted sum of determined times, considering the cost associated with transmitting packets between nodes. This approach captures the inherent complexities and variability present in real-time systems within MDS.

Moreover, the proposed algorithm introduces a metric for comparing diverse algorithms based on process scheduling theorems. This metric evaluates the computational time required to determine a scheduler that effectively adheres to partial order and resource constraints, providing valuable insights into algorithmic efficiency and performance optimization.

$$M_c = \frac{1}{n}\sum_{i=1}^{n} N_{p_i}(t_c) - N_{p_{(i+1)}}(t_c) \tag{3}$$

$$t_c = \max(f_i) - \min(a_i) \tag{4}$$

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.

It's worth noting that the total finish time, as defined in equation (4), encompasses the time taken by the task with the longest duration to complete and the time at which the task that initiated first was started.

Upon expanding the metric expression, a substantial number of terms become null within the telescopic sum, yielding the following equation:

$$M_c = \frac{1}{n}\left(N_{p_1}(t_c) - N_{p_2}(t_c) - N_{p_3}(t_c) + \cdots + N_{p_{(n-1)}}(t_c) - N_{p_n}(t_c)\right)$$
$$M_c = \frac{1}{n}\left(N_{p_1}(t_c) - N_{p_n}(t_c)\right) \tag{5}$$

In the analysis of selecting the task scheduling metric, which serves as a fundamental foundation for the system's core development, the consideration revolves around two key metrics: the total completion time metric and the time-based metric weighted sum of completion times (eq. 5). This choice is driven by the significance of individual time attributes for each term and the necessity to accommodate various types of tasks within the proposed system.

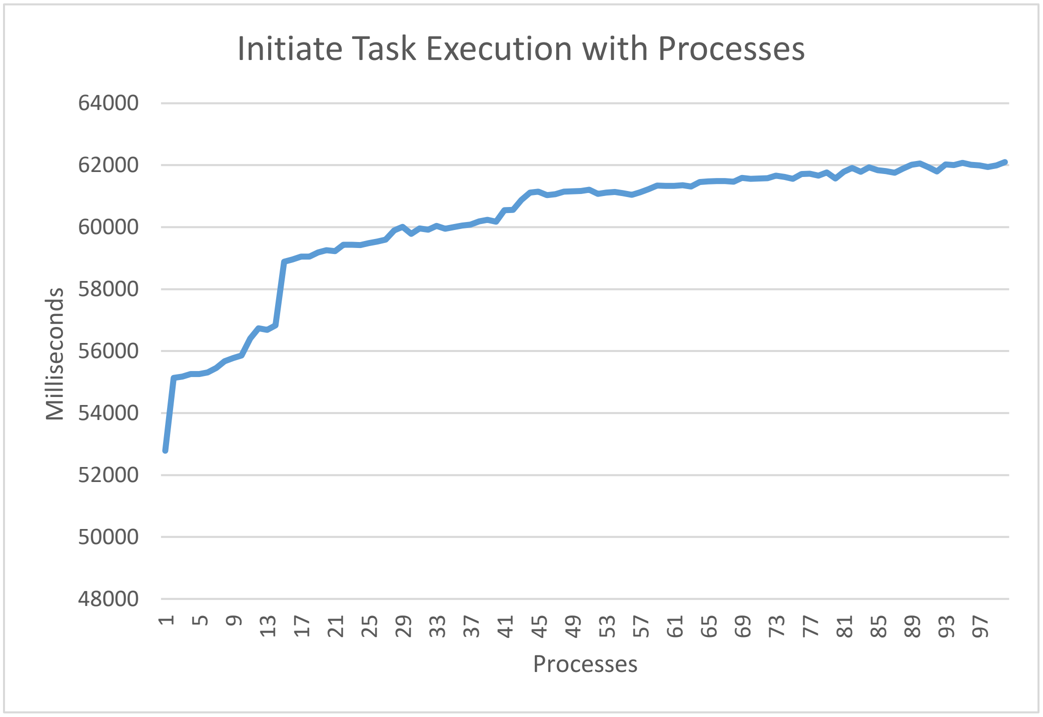

Fig. 3. Initiate Task Execution with Processes: A Fundamental Chronological Framework.

Fig. 3 shows the start of the executions of the tasks from 1 to process 100. The first process enters the term in good condition. As the number of processes and in turn the time to reach your deadline increases.

```
import java.net.*;
import java.io.*;
public class NetworkCommunication {
    static DatagramSocket idServerSocket;
    static InetAddress serverSocketAddr;
    static DatagramPacket receivePacket;
    static class ToNodo {
        String message;
        String ip;
        int port;
    }
   static void sendToMessage(String message, String ip, int port)
throws IOException {
        InetAddress toAddress = InetAddress.getByName(ip);
        byte[] sendData = message.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, toAddress, port);
        idServerSocket.send(sendPacket);
    }
    static String receiveToMessage() throws IOException {
        byte[] receiveData = new byte[1024];
```

```java
        receivePacket = new DatagramPacket(receiveData, receiveData.length);
        idServerSocket.receive(receivePacket);
        return new String(receivePacket.getData(), 0, receivePacket.getLength());
    }
    public static void main(String[] args) {
        try {
            idServerSocket = new DatagramSocket();
            serverSocketAddr = InetAddress.getByName("localhost");
            // Usage examples
            ToNodo messageData = new ToNodo();
            messageData.message = "Hello, world!";
            messageData.ip = "127.0.0.1";
            messageData.port = 12345;
            sendToMessage(messageData.message, messageData.ip, messageData.port);
            String receivedMessage = receiveToMessage();
            System.out.println("Received message: " + receivedMessage);
            idServerSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Figure 4: Approaches for Implementing a P2P Network in a Dynamic Distributed Mobile Environment.

To establish an aerial mobile distributed system that employs a peer-to-peer (P2P) network connection framework, each peer being depicted as a distributed mobile device (refer to Fig. 4). Within this context, innovative real-time scheduling algorithms were devised and executed to assess delay quality within a practical scenario. As an illustration, the plan includes creating and deploying an embedded system featuring drones to facilitate assistance in hard-to-reach environments or during natural disasters.

```
Cliente (Nodo Emisor)                              Red                Servidor (Nodo Receptor)
   |                                         |                                      |
   |-- Crear DatagramSocket --------------   |                                      |
   |-- Preparar mensaje                              |                                      |
   |-- Convertir mensaje a bytes             |                                      |
   |-- Crear DatagramPacket                  |                                      |
   |-- Enviar DatagramPacket ------------>   |                                      |
   |                                         |                                      |
   |                                         |-- Crear DatagramSocket --------------|
   |                                         |-- Preparar buffer                    |
   |                                         |-- Esperar DatagramPacket             |
   |                                         |<-- Recibir DatagramPacket ----------- |
   |                                         |-- Extraer datos y convertir a String |
   |                                         |-- Procesar mensaje                   |
   |                                         |                                      |
```

Figure 5. UDP Network Communication Process in Java

The Fig. 5 depicts the process of UDP network communication using the provided Java code. It illustrates how a message is sent and received between two nodes (a client and a server) through a simplified flowchart. Also, This helps to visualize how the code handles socket creation, datagram sending and receiving, and message processing in a UDP network environment.

The provided Java code implements a basic UDP network communication mechanism, designed to send and receive messages over a network using datagram sockets. The NetworkCommunication class contains essential methods for this functionality. The sendToMessage method, defined at line 12, takes a message, an IP address, and a port number as input parameters. It converts the message into bytes and constructs a DatagramPacket which is then sent to the specified address and port using a DatagramSocket. The method receiveToMessage, starting at line 18, sets up a buffer to store incoming data and waits to receive a datagram packet. Upon receiving a packet, it extracts the data and converts it back into a string for further processing. In the main method, a DatagramSocket is instantiated and bound to the localhost address. The code creates an instance of the nested ToNodo class, populates it with a sample message and network details, and sends the message using sendToMessage. It then waits to receive a message with receiveToMessage, printing the received message to the console. The socket is closed at the end to release network resources. This implementation demonstrates fundamental UDP communication techniques, suitable for applications requiring low-latency message exchanges without guaranteed delivery, like real-time data broadcasting and simple networked applications.

5 Conclusions and Directions for Further Research

This research proposal delved into the study of real-time and consensus task planning algorithms, along with techniques, methods, and algorithms pertinent to distributed mobile systems. The focus was on recent applications and research developments in this field. As a result, a methodology and metric were introduced, poised to effectively address the objectives within a distributed environment.

Furthermore, this endeavor established autonomous models to integrate process planner operations with problem-solving procedures, thereby yielding precise control to calculate delay times in a MDS. The anticipated outcomes revolve around implementing a task scheduling and consensus algorithm to bolster decision-making within groups of mobile agents on a drone flight platform. The aim is to achieve commendable results on real mobile devices. Additionally, a measurement platform for planning algorithms is envisaged to ascertain desired response times. This also extends to supporting theoretical frameworks while analyzing, designing, and developing real-time response algorithms to address natural threats.

Through these results, the ambition is to counteract the domino effect inherent in other real-time task scheduling algorithms, utilizing the proposed Fan algorithm. Its application is anticipated within mobile distributed environments where response time holds paramount importance, particularly in scenarios like global disaster monitoring applications.

In conclusion, algorithm and method simulations were executed in a high-performance computing laboratory. This endeavor yielded crucial data, paving the way for future research endeavors pertaining to metrics and their applications in the realm of MDS. Additionally, the proposed methodologies open avenues for further exploration in the optimization of distributed systems, with a focus on enhancing real-time response and decision-making capabilities in dynamic environments.

Expanding on these findings, future research could delve into the development of more sophisticated algorithms tailored to specific applications within distributed mobile systems. For instance, exploring adaptive algorithms capable of dynamically adjusting task priorities based on changing environmental conditions or resource availability could enhance the robustness and efficiency of real-time task scheduling. Additionally, investigating the integration of machine learning techniques to optimize task allocation and scheduling processes could further improve system performance and adaptability.

Moreover, the proposed measurement platform for planning algorithms presents opportunities for extensive experimentation and validation across various real-world scenarios. By conducting comprehensive tests in diverse environments and under different workload conditions, researchers can refine and validate the proposed methodologies, ensuring their effectiveness and applicability in practical settings.

Furthermore, collaboration with industry partners and stakeholders could facilitate the deployment and real-world testing of the developed algorithms and methodologies. By leveraging industry expertise and resources, researchers can gain valuable insights into practical challenges and requirements, ensuring that the proposed solutions are tailored to meet real-world needs effectively.

In summary, this research lays the groundwork for advancing the state-of-the-art in real-time task scheduling and consensus algorithms for distributed mobile systems. By addressing key challenges and proposing innovative solutions, it contributes to the ongoing efforts to enhance the efficiency, reliability, and adaptability of distributed systems in dynamic environments.

6 Acknowledgments

The authors acknowledge to the people from the Technological development laboratory of Meritorious Autonomous University of Puebla and National Laboratory of Supercomputing of Southeast Mexico which belongs to the CONACyT national laboratories and Technological Development Laboratory FCC-BUAP, for all the technical assistance and the computational resources.

References

1. Baghezza, R., Benidir, M., Chibani, A., Belbachir, A. N., & Amirat, Y. (2021). From offline to real-time distributed activity recognition in wireless sensor networks for healthcare: A review. *Sensors*, *21*(8), 2786.
2. Wazwaz, A., Amin, K., Semary, N., & Ghanem, T. (2024). Dynamic and distributed intelligence over smart devices, Internet of Things edges, and cloud computing for human activity recognition using wearable sensors. *Journal of Sensor and Actuator Networks*, *13*(1), 5. https://doi.org/10.3390/jsan13010005
3. Ageed, Z. S., & Zeebaree, S. R. M. (2024). Distributed systems meet cloud computing: A review of convergence and integration. *International Journal of Intelligent Systems and Applications in Engineering*, *12*(11s), 469-490.
4. Corson, M. S., Macker, J., & Batsell, S. G. (1996). Architectural considerations for mobile mesh networking. In *Proceedings of MILCOM'96 IEEE Military Communications Conference* (Vol. 1). IEEE.
5. Park, V. D., & Corson, M. S. (1997). A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of INFOCOM'97* (Vol. 3). IEEE.
6. Salamun, K., Pavić, I., & Džapo, H. (2021). Dynamic priority assignment in FreeRTOS kernel for improving performance metrics. In *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE.
7. Stankovic, J. A., Ramamritham, K., & Son, S. H. (1995). Implications of classical scheduling results for real-time systems. *Computer*, *28*(6), 16-25.
8. Perale, D., & Vardanega, T. (2021). Removing bias from the judgment day: A Ravenscar-based toolbox for quantitative comparison of EDF-to-RM uniprocessor scheduling. *Journal of Systems Architecture*, *119*, 102236.
9. Murad, S. A., Ahmed, E. A., Qureshi, S. S., & Khan, M. N. A. (2024). SG-PBFS: Shortest gap-priority based fair scheduling technique for job scheduling in cloud environment. *Future Generation Computer Systems*, *150*, 232-242.
10. Bini, E., & Buttazzo, G. C. (2005). Measuring the performance of schedulability tests. *Real-Time Systems*, *30*(1-2), 129-154.
11. Sharma, K., Mahapatra, R., & Kumar, D. (2022). Taxonomy of routing protocols. In *2022 International Conference for Advancement in Technology (ICONAT)*. IEEE.
12. Sengul, C., & Kravets, R. (2006). Bypass routing: An on-demand local recovery protocol for ad hoc networks. *Ad Hoc Networks*, *4*(3), 380-397.
13. Parihar, A. S., & Chakraborty, S. K. (2022). A cross-sectional study on distributed mutual exclusion algorithms for ad hoc networks. In *Pattern Recognition and Data Analysis with Applications*. Springer Nature Singapore.
14. Kshemkalyani, A. D., & Singhal, M. (2011). *Distributed computing: Principles, algorithms, and systems*. Cambridge University Press.

15. Gómez, J. A. H., & Pérez, H. B. (2015). Global scheduling of confined tasks based on consensus. *IEEE Latin America Transactions*, *13*(3), 825-834.
16. Arellano-Vázquez, M., & Benítez-Pérez, H. (2021). An experimental approach to the consensus routing algorithm applied to temporal networks. *IEEE Latin America Transactions*, *19*(9), 1486-1493.
17. Liu, H., Zheng, W. X., & Yu, W. (2021). Continuous-time algorithm based on finite-time consensus for distributed constrained convex optimization. *IEEE Transactions on Automatic Control*, *67*(5), 2552-2559.
18. Vlachou, A., Doulkeridis, C., Nørvåg, K., & Vazirgiannis, M. (2012). Peer-to-peer query processing over multidimensional data. *Springer Science Business Media*.
19. Yang, K., Zhou, L., Wang, W., Zhou, X., & Zhang, Y. (2019). Distributed similarity queries in metric spaces. *Data Science and Engineering*, *4*, 93-108.
20. Joshi, T., Goyal, N., & Ram, M. (2022). An approach to analyze reliability indices in peer-to-peer communication systems. *Cybernetics and Systems*, *53*(8), 716-733.
21. Larios-Gómez, M., Estrada-Ruíz, J. A., & Fernández-Gómez, J. E. (2019). A scheduling algorithm for a platform in real time. In *Supercomputing: 9th International Conference, ISUM 2018, Mérida, Mexico, March 5–9, 2018, Revised Selected Papers, 9*. Springer International Publishing.
22. Kim, B., Lee, S., Park, S., & Kim, H. (2021). Energy-efficient and real-time remote sensing in AI-powered drones. *Mobile Information Systems, 2021,* 1-8.
23. Lim, G. J., Kim, J. H., Cho, J., Gong, Q., & Khodaei, A. (2016). Multi-UAV pre-positioning and routing for power network damage assessment. *IEEE Transactions on Smart Grid*, *9*(4), 3643-3651.
24. Jeong, S., Simeone, O., & Kang, J. (2017). Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning. *IEEE Transactions on Vehicular Technology*, *67*(3), 2049-2063.
25. Nouiri, M., Ghedira, K., & Bouzouita, I. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing, 29,* 603-615.
26. Soria, E., Schiano, F., & Floreano, D. (2021). Distributed predictive drone swarms in cluttered environments. *IEEE Robotics and Automation Letters*, *7*(1), 73-80.
27. Saffre, F., Peet, E., & Smart, D. (2022). Self-swarming for multi-robot systems deployed for situational awareness. In *New Developments and Environmental Applications of Drones: Proceedings of FinDrones 2020*. Springer International Publishing.
28. Schilling, F., Soria, E., & Floreano, D. (2022). On the scalability of vision-based drone swarms in the presence of occlusions. *IEEE Access*, *10*, 28133-28146.
29. Tanenbaum, A. S. (2016). Lessons learned from 30 years of MINIX. *Communications of the ACM*, *59*(3), 70-78.
30. Amadi, C. A. (2018). Design and implementation of a model predictive control on a pixhawk flight controller (Doctoral dissertation, Stellenbosch University).
31. Wang, Z., Zhou, X., Zhu, H., Zhang, Y., & Ma, J. (2022). The application of micro coaxial rotorcraft in warfare: An overview, key technologies, and warfare scenarios. *IEEE Access*, *10*, 40358-40366.
32. Erol, M. R., & Canberk, B. (2020). Intelligent channel utilization discovery in drone-to-drone networks for smart cities. In *Industrial Networks and Intelligent Systems: 6th EAI International Conference, INISCOM 2020, Hanoi, Vietnam, August 27–28, 2020, Proceedings, 6*. Springer International Publishing.
33. Cheng, A. M. K. (2003). *Real-time systems: Scheduling, analysis, and verification*. John Wiley & Sons.
34. Mogili, U. R., & Deepak, B. B. V. L. (2020). An intelligent drone for agriculture applications with the aid of the MAVlink protocol. In *Innovative Product Design and Intelligent Manufacturing Systems: Select Proceedings of ICIPDIMS 2019*. Springer Singapore.
35. Hilburn, B. (2020). An experimental exploration of machine deep learning for drone conflict prediction. In *Machine Learning Paradigms: Advances in Deep Learning-based Technological Applications*, 273-290.
36. Lamport, L. (2022). Deconstructing the bakery to build a distributed state machine. *Communications of the ACM*, *65*(9), 58-66.
37. Jayanti, P., Crain, T., & Zhang, S. (2001). Bounding Lamport's bakery algorithm. In *SOFSEM 2001: Theory and Practice of Informatics: 28th Conference on Current Trends in Theory and Practice of*

Informatics Piešt'any, Slovak Republic, November 24–December 1, 2001 Proceedings 28. Springer Berlin Heidelberg.

38. Larios-Gómez, M., Juárez-Romero, C., Rodríguez-López, E. S., & Aguilar-Calderón, D. (2023). Application of real-time fan scheduling in exploration-exploitation to optimize minimum function objectives. *Applied Computer Science*, *19*(2).
39. Wang, T., Li, X., Cheng, H., & Ma, J. (2018). Efficient multi-task scheduling algorithm in mobile cloud computing with time constraints. *Peer-to-Peer Networking and Applications*, *11*(6), 793-807. https://doi.org/10.1007/s12083-017-0561-9
40. Stavrinides, G. L., & Karatza, H. D. (2019). A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments. *Multimedia Tools and Applications*, *78*, 24639-24655.
41. Amutha, S., Prabu, A., & Kannan, S. (2024). Optimizing cloud resource allocation and load balancing through eco-efficient task scheduling. *International Journal of Intelligent Systems and Applications in Engineering*, *12*(11s), 137-143.
42. Kumar, M. S., & Kumar, G. R. (2023). EAEFA: An efficient energy-aware task scheduling in cloud environment. *EAI Endorsed Transactions on Scalable Information Systems*, *11*(3).
43. Khiat, A., Haddadi, M., & Bahnes, N. (2024). Genetic-based algorithm for task scheduling in fog–cloud environment. *Journal of Network and Systems Management*, *32*(1), 3.