Available online at www.jlls.org

JOURNAL OF LANGUAGE
AND LINGUISTIC STUDIES

ISSN: 1305-578X

Journal of Language and Linguistic Studies, 18(4), 883-895; 2022

Communication Management Through Sockets Developed In Java

Fredys Alberto Simanca Herrera[1] , Lugo Manuel Barbosa Guerrero[2] , Jairo Jamith Palacios Rozo[3]

[1]Universidad Libre. fredysa.simancah@unilibre.edu.co https://orcid.org/0000-0002-3548-0775

[2]Universidad Colegio Mayor de Cundinamarca lmbarbosa@unicolmayor.edu.co Universidad El Bosque barbosalugo@unbosque.edu.co https://orcid.org/0000-0002-0871-8637

[3]Universidad Colegio Mayor de Cundinamarca jjpalacios@unicolmayor.edu.co https://orcid.org/0000-0002-1437-9838

APA Citation:

Herrera, F.A.S., Guerrero, L.M.B., Rozo, J.J.P. , Communication Management Through Sockets Developed In Java, *Journal of Language and Linguistic Studies, 18*(4), 883-895
Submission Date: 06/10/2022
Acceptance Date: 05/12/2022

Abstract

This article presents a general research relationship between communication through the protocol for the control of information transmission and the use of sockets for communication between the socket of one process (client) and the socket of another process (server). Consequently, the objective was to examine code in the java programming language using the client-server architecture. As a method, language codes and task sharing analysis were used in the behavior of requests by simulating the behavior of the server as a service provider and the requesters called clients. The scope of this article is oriented to demonstrate how sockets are a mechanism through which the connection of network communication is made, thus producing knowledge that facilitates another research on the subject.

Keywords Abstraction, applications, java, language, Socket.

Introduction

With the technological development that has been taking place for decades, online communications and services have developed technologies such as sockets that are based on client-server architecture, where one of the services must always be started and alert to the requests that will allow the communication connection, making use of a protocol for this connection.

EMAIL ID : fredysa.simancah@unilibre.edu.co

In general, use is made of a protocol oriented to the established connection, such as the transfer control protocol - TCP, or the user datagram protocol, which is at the transport level.

In this sense, TCP/IP protocols and the socket API are good examples of this: they play a vital role in modern communication and computing, and interoperability between implementations is essential (Bishop et al., 2018). The client-server type architecture is based on message passing systems, where data is represented in the form of messages that implement the sender and receiver process, each time the sender sends a request the receiver processes it by sending a response, generating a sequence of requests as responses that for a distributed system each machine can answer the requests playing the role of server sometimes and client at other times.

The role of the sockets is to establish the connection for sending orders to the server that begins to answer the requests, consisting of an IP address, a communication protocol and a port through which the established service will be accessed, taking into account that the development of collaborative applications for a specific domain requires time and often present errors in the communication services. Collaborative applications are generally built on low-level network abstractions, such as TCP/UDP socket, SIP (Session Initiation Protocol) and RTP (Real-time Transport Protocol) APIs. (Zhang et al., 2010). Abstraction allows for selecting data from a larger set of data and displaying relevant details of the object, allowing for less complex program design. In the case of the Java programming language, abstraction is achieved by using classes and interface design.

Tool-supported security policy definition at a high level of abstraction reduces the complexity of the security policy definition task. Automated transformation of high-level systems and security models into platform-specific artifacts reduces development time (Reznik et al., 2007). Nowadays, developers have created the encapsulated class system to improve socket programming, making inroads into object-oriented programming, and presenting abstract communication protocols based on the TCP/IP protocol. The socket in programming is a communication tunnel that allows two applications to initiate their communication being the basis of Internet protocols for communication with the application of different communication protocols.

One socket (node) listens on a predefined port on an IP, while the other socket responds first to form a connection loop. Then, to effectively achieve this network topology, a JavaScript (JS) node-based environment is used (Rizwan et al., 2021). The study of sockets is important because it allows an understanding of how communication is initiated in the client-server model, making two applications can establish communication between them, the server always listens through a port where the data sent by the client must pass, which connects to the server through the port to send requests, the conversation is programmed employing protocols, although it depends on the operating system used. Unix domain sockets are designed only for local communications, which means that the client and server processes must be on the same host operating system (Shao et al., 2016). An analysis can be carried out on the subject of sockets, finding that a large percentage of research has been carried out on communications in a theoretical way without the presentation of the code in a particular programming language, which is why the main factor that has been established in this article is to analyze part of the code that performs communication in the client-server model. The server, as a process controller, communicates data between the server side (computer) and the client side (microcontroller) or vice versa, RFID card registration application with an RFID USB reader (Figure 1), manages a database and makes calls automatically (D Suprianto et al., 2019).

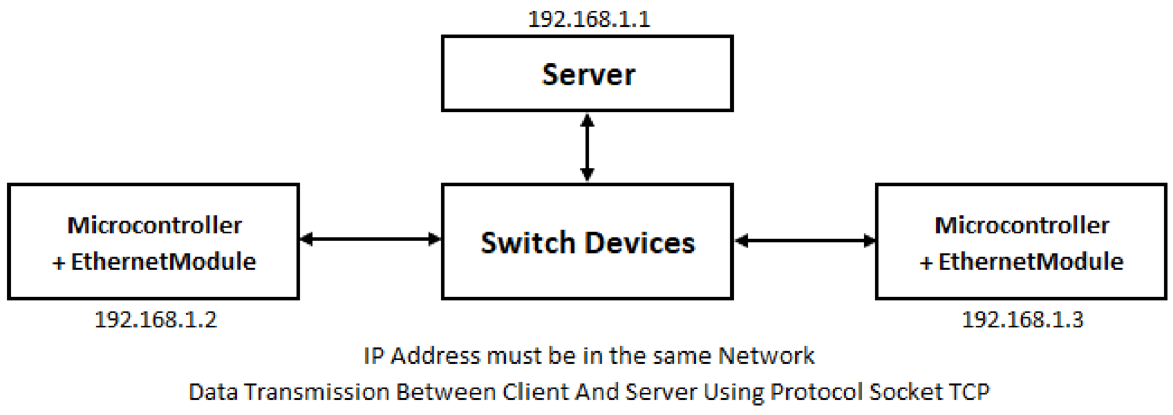
192.168.1.1
Server
Microcontroller + EthernetModule
Switch Devices
Microcontroller + EthernetModule
192.168.1.2
192.168.1.3
IP Address must be in the same Network
Data Transmission Between Client And Server Using Protocol Socket TCP

Figure 1. Data transmission using TCP socket protocol (Suprianto et al., 2019).

Taking into account that sockets present fundamental characteristics in programming languages and especially in Java, communication is also possible between object-oriented languages, for example, a socket designed in PHP can communicate with a socket designed in Java without any inconvenience, communicating between servers regardless of the operating system in which they are running, or like the technology that uses the WebSockets protocol, which has the possibility of initiating an interactive communication session between the client and the server.

The Websocket protocol consists of a client-to-server bidirectional communication that uses HTTP technologies and infrastructures as a transport layer. This protocol facilitates the work of the server, which does not need to initialize different connections for each request from the same client, and uses a TCP connection for client-server and client-server traffic (Saenz et al., 2015). To understand the behavior of sockets in Java. This article presents code made in java that allows the connection between the client that makes the service request and the server that responds to it.

Literature review

No research has considered the use of socket programming with the Java programming language to develop software to implement a client-server service that shows how the server responds to the client's request. The client and server can communicate by writing to and reading from the socket. In socket programming, socket classes are used to build the connection between a client program and a server program (Ahsan et al., 2016).

Java is a programming language that facilitates the creation of this type of protocol and in general communication services due to the portability and security of the language, which defines the syntax and semantics, and also makes use of APIs. The Java Database Connectivity (JDBC) is taken into account here, this API allows the execution of operations on databases from the Java programming language that has a capacity for portability and strength in the management of network applications. It takes advantage of Java's portability and networking capabilities to provide simple mobile software packages, which are composed of a set of bytecode packages together with a configurable and serializable data object (Hulaas et al., 2004).

Java bytecode is inside the .class and is the type of instructions that the Java Virtual Machine (JVM) needs to be able to compile to machine language by the JIT compiler at runtime. JDBC is a Java database connection middleware, which can provide a seamless connection for several very common databases, which can greatly improve the development of programmers' efficiency (Feng and Wu, 2022). Socket services are secure even when maintaining an overhead in the number of flows because they automatically look for dependencies over TCP/UDP, even grouping multiple flows by the same network service.

Network services using TCP are divided into two network service sides: the connection request side and the listening side. The connection requesting side is called the client, and the listening side is called the server. The server opens a fixed listening port and waits for connection requests (Tsubouchi et al., 2022). APIs as software components can be included in the Java platform, allowing for connection and reproduce the functionality that is implemented in the code created by the developer, the ideal of socket work through applications with Java is that it is possible to combine the new code of the developer with the pre-existing code of the APIs in Java. In the object-oriented paradigm with deep use of abstraction, inheritance and generic programming. The Scala API provides a powerful strongly typed functional programming model, which allows the writing of short and expressive code for applications (Poslavsky, 2019). Socket-based methods have many advantages among them low CPU usage, usually in batch processing, sometimes requests to the server are one per day, or even over several days, which allows the server administrator to exclude clients with high overhead, depending on the need for resources.

The technology highly encapsulates the bottom layer of the program and greatly simplifies the program application on the same computer. In addition, it builds a high-performance programming interface for sharing and releasing data between test equipment and automation software through the process of real-time data exchange on different computers connected through the network (Ma and Zhou, 2021).

Socket communication meets the concurrency and security requirements of the client-server model, allowing the communication between client and server to be completed by securely transmitting the flow established between the communicating nodes. Sharing objects with sockets requires that an object is sent to be updated without the open-source code making a reverse reference. The flow established between nodes must be re-established at each communication, an attempt that causes latency in the overall communication or unshared objects (Sharif and Gursoy, 2018). The server contains as many routes as devices can be checked. Each route is invoked by the operator to present information about a device or as a means to perform a security check on the system (Song and Garcia-Valls, 2022). Software developers design encapsulated classes to have a better behavior of socket programming because it is better to be performed at the beginning, the Java programming language facilitates this design through multiple threads using packages such as java.util.concurrent or java.net.ServerSocket. The Java programming language provides comprehensive support for multi-threaded programming through the Subprocess class and the java.util.concurrent package. Since its inception, the language provided multi-threaded programming facilities through the java.lang.Thread class and asynchronous declared methods (Smith and Wells, 2017)

Socket programming is necessary programming for network communications because it is a key part of information gathering, which has been strengthened with object-oriented programming and with the creation and use of multicore processors, in addition to having a socket security protocol (SSL). SSL is developed to allow the web browser and web server to communicate securely by authenticating the server and client. SSL uses a public key to encrypt the transferred data (Suherman et al., 2021). When the data flow is not shared, a large amount of memory will be needed, so that data flows can be passed continuously, allowing the communication to remain active for a long time, although it may seem a problem, this allows the data to flow, be referenced and finally be passed to be delivered.

There is a final obstacle with this type of communication. Unshared objects are only superficially copied and the object flow does not update the values of secondary objects, at the beginning of each message-sending operation (Sharif and Gursoy, 2018). Raising a communication to share objects with sockets requires that an object is sent to be updated without the open source making a reverse reference because the established flow between nodes must be restored at each communication, which can be avoided by programming with the Java language because it has concurrent programming. It should be emphasized that the standard facilities for concurrent programming in Java are inherently thread-based rather than process-based. The current network communication mechanism through the IP stack has significant overhead when used for processes on a single computer system (Smith and Wells, 2017). To understand the socket, it must be clear how TCP/IP protocols, participate in the sending line and publication, thus to assume a sending publication, one must understand the details of this, dividing the information into packets and re-clearing the sending channel, taking into account that the losses manage to reduce the communication traffic by filling the router's queue.

The most significant change experienced by TCP-IP in recent years has been the introduction of TP and TSQ. The cooperative work of these two TCP submodules has a strong impact on the way packets are delivered over the TCP socket, affecting TCP RTT and system latency (Grazia et al., 2021). The traffic control embedded in a TCP/IP protocol is a typical closed loop. The flow of packets issued by a sender is controlled by the probability of loss observed at the routers and reported to the sender with some delay. Losses decrease traffic, and their lack increases it, in the case of a passive router, losses occur when the router queue is full (Marek et al., 2021). For TCP/IP sockets, there is no synchronization between reading and write requests on both sides of the connection. After migration, the msocket library creates a new socket (during restoration), registers it and asks the mirror to deregister the old one (Bubak et al., 2001).

Materials and methods

This research focused on the simulation of a client-server model to analyze the behavior of the abstraction of sockets and their behavior in communications using a software design in the Java programming language. The software was designed to manage the process of requesting a client to the system and when obtaining the response from the server, initiate the communication through a chat. Taking into account that Java only supports socket communication, using the loopback system, which allows network-based communication mechanisms to make use of sockets and TCP/IP protocols. The Java programming language provides integral support for multiprocess programming and in the case of the research, several native Java classes were used that provide multiple subprocess programming facilities through them, managing to create objects that could interact with each other by sending messages to achieve communication. The java.io package has to be imported, which is the one in charge of the management of both input and output operations.

DataInputStream

Java provides the DataInputStream class that allows reading data of primitive type because it has a single constructor that takes an object of the InputStream class to create an object which links it to another FileInputStream object, so it can read from a flat file, for the example, it is called test.txt:

```
FileInputStream fileIn=new FileInputStream("prueba.txt");
DataInputStream input=new DataInputStream(fileIn));
```

The advantage of designing with a language of the object-oriented programming paradigm is that it is possible to count on structural reflection, which is considered key in dynamic languages and especially in Java, which allows the designing program to obtain internal information of any class through the API Reflection. This helps in the execution because there is the direct management of the internal properties and methods of the objects. In the Structural reflection, in case the program structure is modified, the changes will be reflected at runtime. The ability to read class structures (e.g., Java) and modify them (e.g., Python) are two examples of structural reflection (Lagartos et al., 2019).

DataOutputStream

This class is useful when it is required to write data of primitive type but portable, creating an object of this class, but linking it to an object to be able to write to a file. The DataOutputStream class is found in the java.io package.

This class is used whenever you want to write large amounts of data to the output stream. With the help of the New operator the object is created and it can be of any type, its structure is:

```
DataOutputStream out = new DataOutputStream(new FileOutputStream("prueba.txt"));
```

IOException

It is used in the Java programming language to catch the exceptions that are generated in the block of code delimited by the try and catch statements that have the function of handling the code fragments that tend to present failures for example when a null value is typed or a data type that does not correspond. The try-catch makes sure that, even if the task that is being executed fails, the program continues executing and does not stop, when the code that has the try fails immediately the catch will be executed so that the program continues executing. A simple example can be seen in the following code where an object to assign the port is created:

```
try {
    ServerSocket skServidor = new ServerSocket( PORT);
    System.out.println("Communication with the port is established" + PORT);
```

java.net.Socket

Sockets are a mechanism that allows establishing a link between two programs that are executed independently of who is the client and who is the server. Therefore, the Socket class that belongs to the java.net package is an independent implementation, it is worth clarifying that java has two classes that handle communications, one is the socket that allows implementing the connection as a client and the other is the ServerSocket that, as part of its name indicates, implements the communication from the server. Through the socket sentence, it is created to be able to make a connection in the communication as a client, also through the writeUTF sentence, the first message is sent.

```
for (int numCli = 0; numCli < 15; numCli++) {
Socket skCliente = skServidor.accept();
System.out.println("Servicio iniciado " + numCli);
OutputStream aux = skCliente.getOutputStream();
DataOutputStream flujo= new DataOutputStream( aux );
flujo.writeUTF( "Conexión preparada… inicia comunicación" + numCli );
skCliente.close();
}
```

java.util.logging.Level

It acts as an interface for the chronological control in the data logging, using the basic logging and tracking method, counting on a chronological logging API, the events are created and the information is automatically filled in. One of the most used libraries for the implementation of logging is Log4j because it allows having more advanced functionalities even than the java.util.logging.Level, but this last one is enough for the communication.

Both client and server programs must call the following libraries:

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
```

Results and discussion

The tests were performed on Intel(R) Core (TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz, with Windows 11 Home Single Language, 64 bits. The programming language was Java for Windows - Version 8 Update 333, in the whole process of socket communication, two applications were designed, one for the client and another for the server, which when executed began to send the communication messages. When the communication presented execution failures, the exceptions and the failure in the server socket started to be executed, showing the following lines:

```
java.net.BindException: Address already in use: bind
at java.base/sun.nio.ch.Net.bind0(Native Method)
at java.base/sun.nio.ch.Net.bind(Net.java:552)
at java.base/sun.nio.ch.Net.bind(Net.java:541)
at java.base/sun.nio.ch.NioSocketImpl.bind(NioSocketImpl.java:643)
at java.base/java.net.ServerSocket.bind(ServerSocket.java:395)
at java.base/java.net.ServerSocket.<init>(ServerSocket.java:281)
at java.base/java.net.ServerSocket.<init>(ServerSocket.java:172)
at cliente_servidor.Servidor.main(servidor.java:25)
```

However, since there was communication between the client and the server, messages were displayed without any inconvenience, and the sending of requests was carried out in several tests, with a simulation of 15 clients in each one, having acceptable response times as shown in Table 1.

Table 1. Response times to requests serving clients at the same time.

Request	**Response time**	**milliseconds**
Number of customers: 1	BUILD SUCCESSFUL (total time: 0 msec	0.00000000000
Number of customers: 2	BUILD SUCCESSFUL (total time: 0.000000001 msec	0.00000000100
Number of customers: 3	BUILD SUCCESSFUL (total time: 0.0000000011 msec	0.00000000110
Number of customers: 4	BUILD SUCCESSFUL (total time: 0.0000000012 msec	0.00000000120
Number of customers: 5	BUILD SUCCESSFUL (total time: 0.0000000013 msec	0.00000000130
Number of customers: 6	BUILD SUCCESSFUL (total time: 0.0000000014 msec	0.00000000140
Number of customers: 7	BUILD SUCCESSFUL (total time: 0.0000000015 msec	0.00000000150
Number of customers: 8	BUILD SUCCESSFUL (total time: 0.0000000016 msec	0.00000000160
Number of customers: 9	BUILD SUCCESSFUL (total time: 0.0000000017 msec	0.00000000170
Number of customers: 10	BUILD SUCCESSFUL (total time: 0.0000000018 msec	0.00000000180
Number of customers: 11	BUILD SUCCESSFUL (total time: 0.0000000019 msec	0.00000000190
Number of customers: 12	BUILD SUCCESSFUL (total time: 0.000000002 msec	0.00000000200
Number of customers: 13	BUILD SUCCESSFUL (total time: 0.0000000021 msec	0.00000000210
Number of customers: 14	BUILD SUCCESSFUL (total time: 0.0000000022 msec	0.00000000220
Number of customers: 15	BUILD SUCCESSFUL (total time: 0.0000000023 msec	0.00000000230

These times allow seeing the behaviors of the sockects when starting communications, serving clients simultaneously, the link functions between the clients and the server showed excellent performance. The remaining functions were subsequently tested and their performance was compared to that of socket-based communication, with the tests examining the throughput and latency behavior of the programs when transferring data in real use. These results demonstrate the improvement in response time, which is logically linked to the performance of the machines acting as clients making requests to the server. Figure 2 shows more clearly the results on the response time according to the number of clients, allowing a better comparison of the response time to the requests.

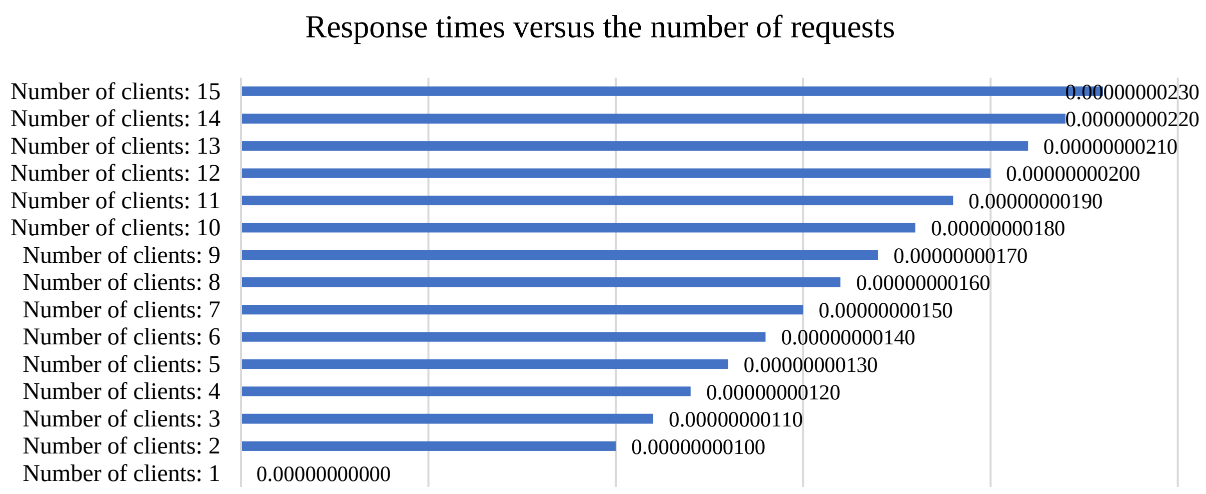

Figure 2. Response times vs. the number of requests at the same time.

Proving that the use of sockets designed in the java programming language facilitates communication in a client-server model is perfect for managing chats between other systems, the sockets have the strength in the interconnection through the network, as seen in the code, only with the IP address or server name and port determination.

```
public class Cliente {
static final String HOST = “localhost”;
static final int PUERTO=80;
public Cliente( ) {
try{
Socket skCliente = new Socket( HOST , PUERTO );
InputStream aux = skCliente.getInputStream();
DataInputStream flujo = new DataInputStream( aux );
System.out.println( flujo.readUTF() );
skCliente.close();
```

Table 2 shows the percentage of the average performance in the response time rate to requests according to the number of simultaneous clients, without taking into account the size of the message, only the time at which the request is initiated and the time it takes for the first response.

Table 2. Percentage of Response Times.

Request	**milliseconds**	**Percentage**
Number of clients: 1	0.00000000000	0.00%
Number of clients: 2	0.00000000100	4.33%

Number of clients: 3	0.00000000110	4.76%
Number of clients: 4	0.00000000120	5.19%
Number of clients: 5	0.00000000130	5.63%
Number of clients: 6	0.00000000140	6.06%
Number of clients: 7	0.00000000150	6.49%
Number of clients: 8	0.00000000160	6.93%
Number of clients: 9	0.00000000170	7.36%
Number of clients: 10	0.00000000180	7.79%
Number of clients: 11	0.00000000190	8.23%
Number of clients: 12	0.00000000200	8.66%
Number of clients: 13	0.00000000210	9.09%
Number of clients: 14	0.00000000220	9.52%
Number of clients: 15	0.00000000230	9.96%

The results obtained confirm that communications with the management of sockets in the java programming language allow understanding of how they work even at a low level, making it clear that the server must be started first so that the clients can then have answers to the requests. The average time between request and response generated by Sockets demonstrates better performance than protocols for data transmission such as Transmission Control Protocol - TCP or Internet Protocol - IP or protocols associated with the Internet such as Post Office Protocol - POP, Simple Mail Transfer Protocol - SMTP or Internet Message Access Protocol - IMAP. As a result, making use of socket structures designed in the java programming language provides a more efficient performance, with its very low request and response times, in addition to having another benefit which is the writing of both incoming and outgoing data flows. Another step to take into account is to share the memory for the use of the sockets at the same time as the operating system to improve the times of the requests.

Conclusions

The research has demonstrated the inefficiencies of socket communication for inter-process communication in Java and has provided an alternative that uses Microsoft's native Inter-Process Communication (IPC) facilities, the Universal Windows Platform (UWP) and Win32 applications.

Sockets can improve communication by making use of service applications because they allow handling services that accept and return primitive data property code that can continue to be processed in the background or as a process within the foreground application, which improves processes where near real-time latency is not required.

The ultimate goal of the research is to provide a new efficient communication tool with high program portability, in addition to having a data flow architecture that allows measuring the flow of input data through the system, as well as emphasizing the incremental transformation of data in successive requests to the server.

References

Ahsan, M., Haider, J., McManis, J. y Hashmi, M. (2014). Developing intelligent software interface forwireless monitoring of vehicle speed andmanagement of associated data. Special Issue: Selected Papers from the 9th International Symposium onCommunications Systems, Networks and Digital Signal Processing (CSNDSP 2014) IET Wirel. Sens. Syst., 2016, Vol. 6, Iss. 3, pp. 90–9990 & The Institution of Engineering and Technology 2016

Bishop, S., Fairbairn, M., Norrish, M., Sewell, P. y Smith, M. (2018). Engineering with Logic: Rigorous Test-Oracle Specification and Validation for TCP/IP and the Sockets API. Journal of the ACM, Vol. 66, No. 1, Article 1. Publication date: December 2018

Feng, P. y Wu, Q. (2022). Digital Teaching Management System Based on Deep Learning of Internet of Things. Hindawi. Mobile Information Systems. Volume 2022, Article ID 3414935, 11 pages. https://doi.org/10.1155/2022/3414935

Grazia, C., Klapez, M. y Casoni, M. (2021). The New TCP Modules on the Block: A Performance Evaluation of TCP Pacing and TCP Small Queues. IEEEAccess. Digital Object Identifier 10.1109/ACCESS.2021.3113891

Hulaas, J., Binder, W. y Di Marzo, G. (2004). Enhancing Java Grid Computing Security with Resource Control. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)
3270, pp. 30-47

Lagartos, I., Redondo, J. y Ortin, F. (2019). Efficient runtime metaprogramming services for Java, Journal of Systems and Software, Volume 153, 2019, Pages 220-237, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2019.04.030.

Ma, H. y Zhou, Q. (2021). Application of computer network virtual instrument technology in the development of a construction machinery remote detection system. International. Journal of Mechatronics and Applied Mechanics, 2021, Issue 10, Vol. I

Marek, D.; Domanski, A., Domanska, J., Szyguła, J., Czachórski, T. y Klamka, J. (2021). Diffusion Model of a Non-Integer Order PIγ Controller with TCP/UDP Streams. Entropy 2021, 23, 619. https://doi.org/ 10.3390/e23050619

Marian Bubak, Dariusz Żbik, Dick van Albada, Kamil Iskra, Peter Sloot, "Portable Library of Migratable Sockets", Scientific Programming, vol. 9, Article ID 126087, 12 pages, 2001. https://doi.org/10.1155/2001/126087

Poslavsky, S. (2019). Rings: An efficient Java/Scala library for polynomial rings. Computer Physics Communications. Volume 235, February 2019, Pages 400-413
ISSN 0010-4655, https://doi.org/10.1016/j.cpc.2018.09.005

Reznik, J., Ritter, T., Schreiner, R. y Lang, U. (2007). Model Driven Development of Security Aspects. Electronic Notes in Theoretical Computer Science Volume 163, Issue 2, 16 April 2007, Pages 65-79

Rizwan, R.; Arshad, J.; Almogren, A.; Jaffery, M.H.; Yousaf, A.; Khan, A.; Ur Rehman, A.; Shafiq, M. Implementation of ANN-Based Embedded Hybrid Power Filter Using HIL-Topology with Real-Time Data Visualization through Node-RED. Energies 2021, 14, 7127. https://doi.org/10.3390/en14217127

Saenz, J., Esquembre, F., Garcia, F., De la Torre, L. y Dormido, S. (2015). An Architecture to use Easy Java-Javascript Simulations in New Devices**Sponsor and financial support acknowledgment goes here. Paper titles should be written in uppercase and lowercase letters, not all uppercase., IFAC-PapersOnLine, Volume 48, Issue 29, 2015, Pages 129-133, ISSN 2405-8963.
https://doi.org/10.1016/j.ifacol.2015.11.225.

Shao, Y., Ott, J., Jia, Y., Qian, Z. y Mao, Z. (2016). The Misuse of Android Unix Domain Sockets and Security Implications. CCS '16: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications SecurityOctober 2016 Pages 80–91
https://doi.org/10.1145/2976749.2978297

Sharif, H. y Gursoy, O. (2018). Parallel Computing for Artificial Neural Network Training using Java Native Socket Programming. Periodicals of Engineering and Natural Sciences Vol. 6, No. 1, February 2018, pp. 1 – 10
Disponible en línea en: http://pen.ius.edu.ba

Smith, D.G., and Wells, G.C. (2017). Interprocess communication with Java in a Microsoft Windows Environment. South African Computer Journal 29(3), 198–214. https://doi.org/10.18489/sacj.v29i3.500

Song, L., García-Valls, M. (2022). Improving Security of Web Servers in Critical IoT Systems through Self-Monitoring of Vulnerabilities. Sensors 2022, 22, 5004.
https://doi.org/10.3390/s22135004

Suherman, S., Deddy Dikmawanto, D., Hasan, S. y Al-Akaidi, M. (2021). Embedding the three pass protocol messages into transmission control protocol header Indonesian Journal of Electrical Engineering and Computer Science. Vol. 22, No. 1, April 2021, pp. 442~449
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v22.i1.pp442-449

Suprianto, D., Wibowo, D., Setiawan, A. y Agustina, R. (2019). Integrated application design to facilitate effective and safe student pick-up process by utilizing microcontrollers, socket TCP, and client-server database. 4th Annual Applied Science and Engineering Conference. Journal of Physics: Conference Series.

Tsubouchi, Y., Furukawa, M. y Matsumoto, R. (2022). Low Overhead TCP/UDP Socket-based Tracing for Discovering Network Services Dependencies Journal of Information Processing Vol.30 260–268 (Mar. 2022)
DOI: 10.2197/ipsjjip.30.260

Zhang, C., Sadjadi, M., Sun, W., Rangaswami, R. y Deng, Y. (2010). A user-centric network communication broker for multimedia collaborative computing. Multimed Tools Appl (2010) 50:335–357. DOI 10.1007/s11042-009-0385-6