

**DATA TRANSFER USING TCP SOCKET OVER HTTP
APPLICATION**

Project report submitted in partial fulfillment of the requirement for
the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

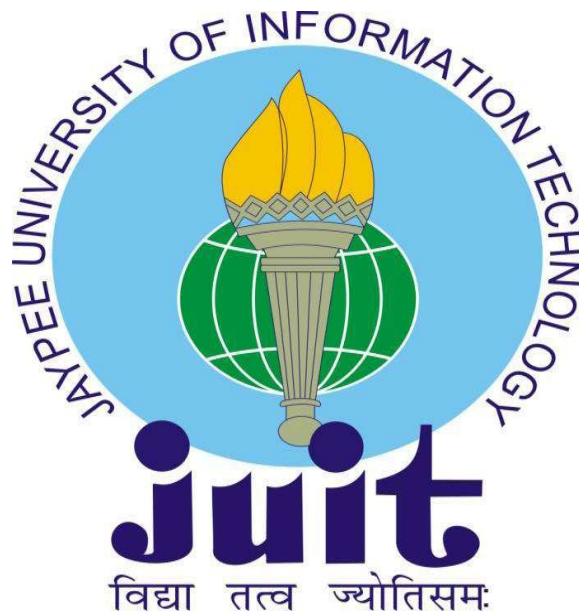
By

Kunal S Bhandari (191205)

Under the supervision of

Dr. Pradeep Kumar Gupta

to



Department of Computer Science & Engineering and Information
Technology

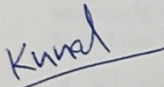
**Jaypee University of Information Technology Waknaghat,
Solan-173234, Himachal Pradesh**

Candidate's Declaration

I hereby declare that the work presented in this report entitled “ **Data Transfer Using TCP Socket Over HTTP Application** ” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Wanknaghat is an authentic record of my own work carried out over a period from August 2015 to December 2015 under the supervision of (**Dr. Pradeep Kumar Gupta**) (Associate Professor and Computer Science).

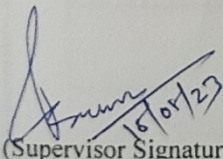
I also authenticate that I have carried out the above mentioned project work under the proficiency stream **Computer Science and Engineering**.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.



Kunal S Bhandari 191205

This is to certify that the above statement made by the candidate is true to the best of my knowledge.



(Supervisor Signature)

Supervisor Name: Dr. Pradeep Kumar Gupta

Designation: Associate Professor

Department name: Computer Science

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 15/5/23

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: KUNAL BHANDARI Department: CSE Enrolment No 191205

Contact No. 9990262296 E-mail. bhandarik667@gmail.com

Name of the Supervisor: Dr. Pradeep KUMAR GUPTA

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters):
DATA TRANSFER USING T.C.P SOCKET OVER HTTP APPLICATION

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 45
- Total No. of Preliminary pages = 8
- Total No. of pages accommodate bibliography/references = 3

Kunal
(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found Similarity Index at 14%.....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[Signature]
(Signature of Guide/Supervisor)

[Signature]
Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String 		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

I would like to thank and express our gratitude to our Project supervisor Dr. Pradeep Kumar Gupta for the opportunity that he provided us with this project “Data Transfer Using TCP Socket Over HTTP Application”. The outcome would not be possible without his guidance. This project taught me many new things and helped to strengthen concepts of Computer Networks and Socket Programming . Next, I would like to express my special thanks to the Lab Assistant for cordially contacting us and helping us in finishing this project within the specified time. Lastly, I would like to thank my friends and parents for their help and support.

TABLE OF CONTENTS

Chapter 1 : INTRODUCTION

Introduction	1
Problem statement	2
Objective.....	2
Methodology.....	3
Organization	6

CHAPTER 2 : LITERATURE SURVEY

Computer Network Architecture	8
Data Transfer protocols	10
TCP Socket Programming	11

CHAPTER 3: SYSTEM DEVELOPMENT 13

CHAPTER 4: PERFORMANCE ANALYSIS

1.1: Comparisons and Results	37
------------------------------------	----

CONCLUSIONS

Conclusion.....	42
-----------------	----

REFERENCES43

LIST OF FIGURES

Figure 1:TCP MODEL VS OSI MODEL AND THEIR PROTOCOLS IN EACH LAYER	1
Figure 2: ARCHITECTURE OF WEB SOCKET CONNECTION	4
Figure 3: CONTAINER ARCHITECTURE.....	5
Figure 4:ONE TO ONE RELATION OF ITEMS IN DATABASE.....	13
Figure 5: RELATION OF EACH VALUES AND USER DATABASE DESIGN.....	14
Figure 6 : WORKFLOW OF SERVER AND DB	15
Figure 7 :REQUEST RESPONSE CYCLE OF THE SYSTEM.....	16
Figure 8 : DATABASE VALIDATION SCHEMA.....	23
Figure 9 : FACTORY DESIGN PATTERN.....	26
Figure 10 : DOCKER FILE.....	28
Figure 11 : DOCKER COMPOSE YAML FILE	29
Figure 12 : DOCKER COMPOSE-DEBUG YAML FILE	30
Figure 13 : TCP SOCKET ARCHITECTURE	31
Figure 14 : CLIENT - SERVER TCP SOCKET TRANSMISSION DESIGN	32
Figure 15 : FUNCITON TO FETCH OPPONENT SOCKET ID.....	33
Figure 16 : FUNCTION TO CREATE SOCKET OBJECT	33
Figure 17 : SOCKET FOR USER STATE.....	34
Figure 18 : SOCKET FOR STATUS CHECK	34
Figure 19 : SOCKET FOR STATE OF BOARD.....	35
Figure 20 : SOCKET FOR DISCONNECTION	36

LIST OF GRAPHS

GRAPH 1: PARAMETER I ANALYSIS GRAPH.....	39
GRAPH 2:PARAMETER II ANALYSIS GRAPH.....	40
GRAPH 3:PARAMETER III ANALYSIS GRAPH	41

ABSTRACT

The application described in this abstract is a bidirectional communication tool that uses Socket.IO and Node.js to facilitate real-time data transfer over both HTTP and TCP sockets. This application enables seamless communication between clients and servers, allowing for instantaneous exchange of data and updates.

By leveraging the power of Socket.IO, this application provides a reliable and efficient means of bidirectional communication, with support for multiple connections and real-time event-driven communication. Users can send messages, notifications, and other data in real-time, with near-instantaneous delivery and response times. Furthermore, this application supports both HTTP and TCP sockets, allowing for flexible communication options based on the specific needs of the user. This makes it an ideal solution for a wide range of use cases, including chat applications, real-time dashboards, and online gaming platforms. Overall, this bidirectional communication tool provides a powerful and versatile solution for anyone seeking to implement real-time communication capabilities in their web or mobile applications. With support for both HTTP and TCP sockets, and the ability to handle large numbers of concurrent connections, it offers a highly scalable and efficient means of data transfer that is both reliable and responsive.

CHAPTER 1 : INTRODUCTION

1.1 Introduction

The Hypertext Transport Protocol (HTTP) and the Transmission Control Protocol (TCP) are computer protocols that facilitate data transport. HTTP is a protocol for responding to requests, allowing clients to interact and transport hypertext on the worldwide web (WWW). The protocol is still one of the most used ways to access the Internet, allowing users to interact with online assets, including HTML documents, by sending texts between consumers and servers. It's basically used to load online pages via hypertext links. It is the foundation of the World Wide Web and defines the message format used by Internet browsers and Internet servers to interact. It also specifies how a web browser should handle a specific web request. It is a stateless protocol, which implies that the receiver does not keep session information from prior requests. Although the core of HTTP is stateless, it is not session-less because HTTP cookies allow for the utilization of state sessions.

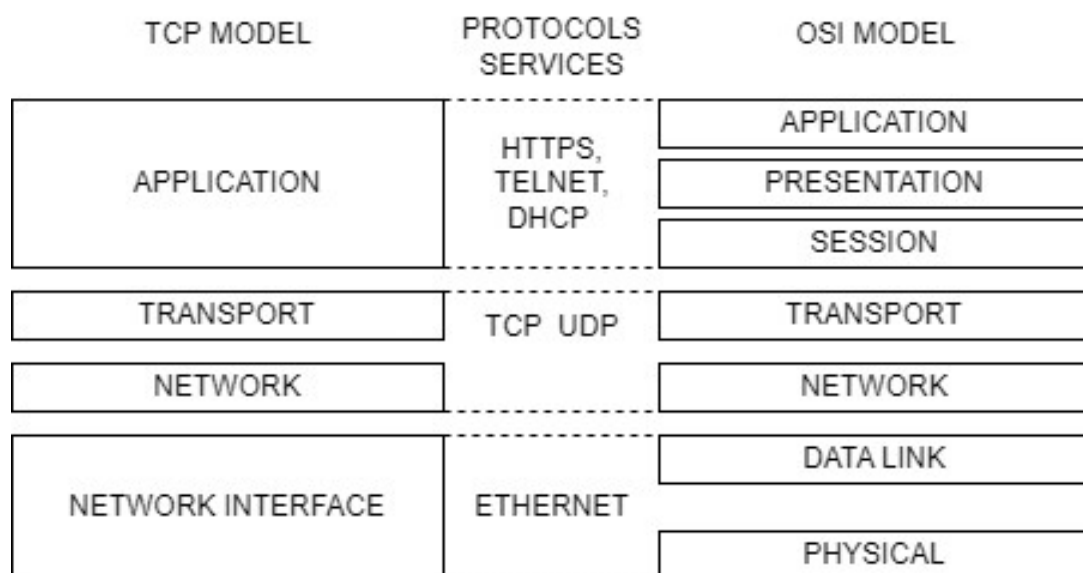


FIG 1. TCP MODEL VS OSI MODEL AND THEIR PROTOCOLS IN EACH LAYER .

TCP, which signifies Transmission Control Protocol, is a form of communication protocol that allows application programs and computer devices to send and receive

data and/or messages over networks. It is a stateful protocol that is included in the Internet Engineering Task Force (IETF) specifications.

TCP protocol specifies how to set up and up a network connection via which data may be transferred. It also selects how to divide application data into network-transferable packets and assures end-to-end data transmission. TCP transmission is dependable, and secure, and ensures the integrity of information delivered over an internet connection, regardless of its size. TCP is located at Layer 4 of the Open Systems Interconnection (OSI) Model and is used in conjunction using the Internet Protocol. IP is a primary communications protocol that specifies how data should be transmitted from one network to the next via the Internet.

1.2 Problem Statement

When you want to create a real time bidirectional communication between the client and the web server there are several communication protocols to choose from while developing apps: the Hypertext Transfer Protocol , the Web-socket and WebRTC. To send and receive data correctly, equipment on either side of the exchange of Protocol guidelines must be accepted and followed by information. Protocol support in communication can be built into either software or hardware. Systems and other equipment would be unable to interact with one another without network protocols. As a result, a smaller amount networks would be able to remain functioning, with the exception of speciality networks built around a certain design, and the internet as we know it would go away. Almost all communication end users rely on network protocols to connect. Different protocols follow different rules, and it is critical to understand each protocol's strengths and limitations. In this instance, we must select the best protocol for data transfer between clients and servers in order to accommodate real-time data transmission .

1.3 Objectives

Some applications necessitate a peer-to-peer connection with minimal latency and high transmission of data, and they are willing to accommodate some packet (information) losses. Other apps may query the server as needed and do not need to communicate with a different peer. A protocol is a collection of rules that regulate how data is transferred between devices in computer networking. The protocol

specifies the communication rules, grammar, semantics, synchronization, and error recovery mechanisms. A network protocol is a set of rules that determines how data is sent among devices on an identical network. It essentially allows connected devices to communicate with one another despite differences in their internal functions, structure, or design. Network protocols provide simple contact with people globally and so play a vital role in modern digital communications.

More than only trained network engineers and IT experts rely on network protocols. Every day, billions of individuals use network protocols, whether they realize it or not. You use network protocols every time you access to the internet. Though you may not comprehend how network protocols function or how frequently you encounter them, they are necessary for any usage of the World Wide Web or electronic communication. The communication and data requirements of a multiplayer online game play, chat app, blogging website, multimedia galleries app, and video calling software vary. Other factors to consider when developing a video streaming solution can be found in this report.

1.4 Methodology

1.4.1 Hypertext Transfer Protocol

Hypertext Transfer Protocol, also known as HTTP, is a networked, cooperative, and multimedia information system application protocol. HTTP is the World Wide Web's data transfer protocol. Hypertext is a type of organized text that employs logical linkages (hyperlinks) between text-containing nodes. The HTTP protocol is used to exchange or transport hypertext. Servers and clients communicate via HTTP by exchanging distinct messages. Requests are information sent by the client, while replies are messages delivered by the server. These messages are transmitted as plain text via a TCP connection. They can also be protected with TLS and transmitted via the HTTPS protocol. A client is often a browser for the internet or program running on a user's phone or machine, but it might potentially be anything, such as a crawling script. Requests made via HTTP can only be sent from a client's computer to the server. The server cannot start contact with the user; it can only reply to requests.

1.4.2 Web Socket

Web Socket is a protocol that allows a client and the server to communicate in both directions. It's an attractive option for real-time data applications including chat apps, multiplayer games, and live information streaming. The Web Socket protocol is a bidirectional protocol that is commonly used in client-server channels of communication. It is bidirectional in nature, which means that communication takes place between the client and the server.

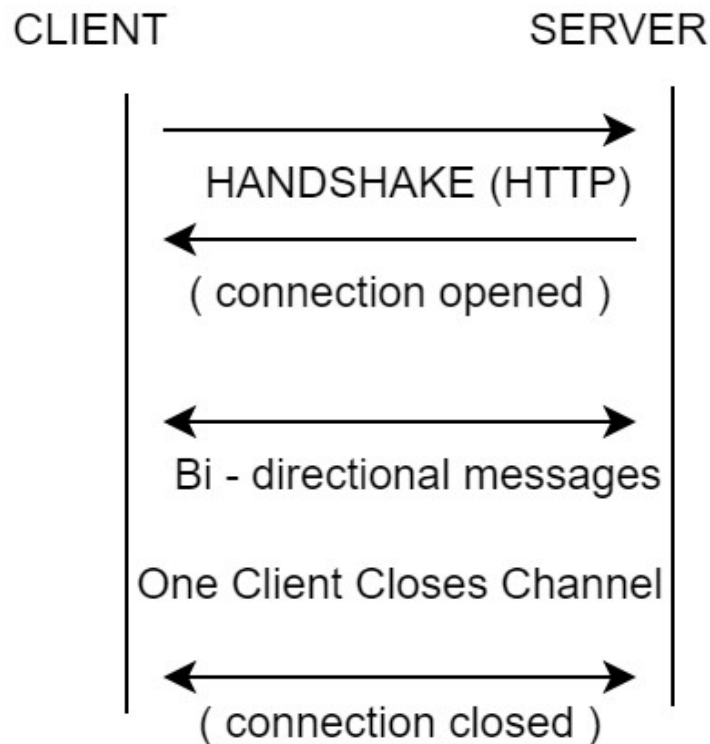


FIG 2. ARCHITECTURE OF WEB SOCKET CONNECTION .

The Web Socket-based relationship remains active until one party disconnects it. When one party stops contact, the other side is unable to share information because the link is immediately severed at its front. Web Socket necessitate HTTP capabilities to create a connection. In terms of functionality, it provides the backbone for advanced web app development when it comes to smooth data streaming and other unsynchronized traffic.

1.4.3 Containers

Containers are packages of software that have all of the necessary components to run in any environment. Containers replicate the operating system and enable it to run anywhere, whether an internal data centre to a cloud service to a developer's the

machine. Our development teams can now move quickly, distribute software effectively, and function at an unparalleled scale thanks to containerization.

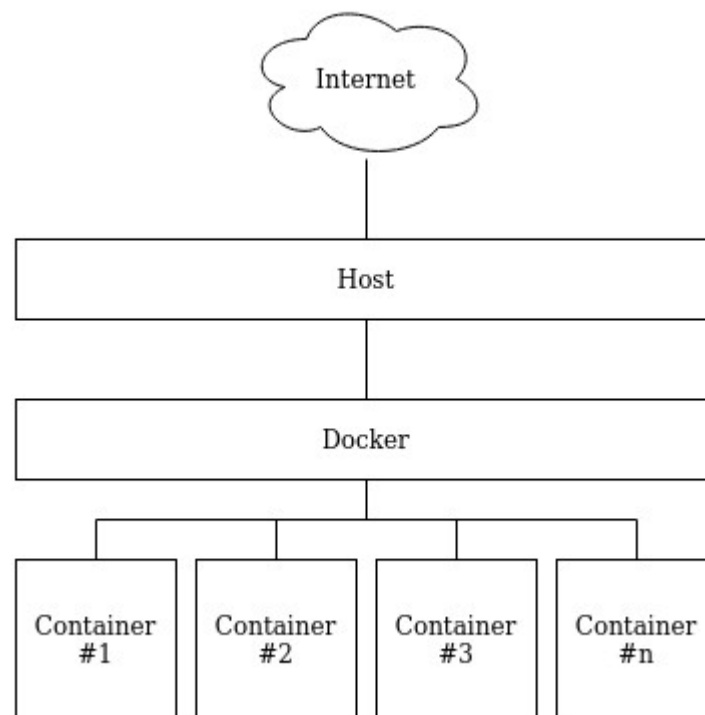


FIG 3. CONTAINER ARCHITECTURE .

Containers are executable software units in which application code, together with its frameworks and interdependence, is packed in standard ways so that the code may be executed anywhere. Containers accomplish this through a type of operating systems (OS) virtualization in which features of the kernel of the OS are used to isolate handles and control the amount of CPU, memory, and disc that those processes can access. Containers are compact, quick, and portable because, unlike virtual machines, they do not require an additional operating system in each instance and may instead rely on the host OS's capabilities and resources.

1.4.4 TCP Protocol

TCP, or Transmission Control Protocol, is a network communication protocol that allows application programme and hardware components to exchange messages. Its goal is to send packets across the internet and ensure that messages and other information are delivered correctly across networks. TCP is a key protocol that establishes Internet laws and has been incorporated in the Internet Engineering Task

Force (IETF) specifications. It is a commonly used protocol for digital communication over networks that ensures data delivery from start to end. Transmission Control Protocol (TCP) connections have been employed to link the server and client processes. The server code is executed first, which establishes a port and waits for incoming client connection requests.

In networking, The User Data-gram Protocol (UDP) is a protocol that, like TCP, is used to establish low-latency connections between programme and shorten transmission times. Because it incorporates missing or improperly formed packets and protects data transfer with restrictions such as acknowledgment of receipt connection initiation, and flow control, TCP is an expensive network utility. UDP is less reliable but less expensive since it lacks defective connection and packet the sequencing process, as well as signaling a destination before transmitting data. As a result, it is an excellent solution for time-critical applications which include DNS search, Voice over Internet Protocol (VoIP), and video streaming.

1.5 Organization

Chapter 1: Introduction

Various aspects of the project are dealt with in this section, which includes a synopsis of the assignment, the approach that was used, an explanation of the issue discussed by the project, as well as the goal of the project.

Chapter 2: Literature Survey

This project's literature review part discusses the evaluated resources as well as the topics that have been investigated and recognized.

Chapter 3: System Design and Development

Within this section, we are going to look over project analysis and system design implementation. We will describe the algorithms employed and share project snapshots.

Chapter 4: Experiment and Result Analysis

This portion of the project shows the results of the analysis by displaying and comparing the findings in snapshots. It also has the ability to show multiple outputs. The part also discusses how the outcomes were acquired and what they indicate for the project's achievement.

Chapter 5: Conclusion

This component of the study ends with the present project and discusses potential additional study and development opportunities.

CHAPTER 2 : LITERATURE SURVEY

The numerous advancements in the field of networking and socket programme were covered in this part. The effectiveness of the system and socket programming methodologies, as well as the outcomes of their testing on the specified architecture

2.1 Computer Networking Architecture

- It has been found that the types of network assaults are equally varied as the systems they target.[11] Technically skilled intruders might have been fascinated with attacking the protocols that are used for safe communication across networking devices, whether the attacks are purposeful or accidental . This review discusses how, despite very high-security measures, extremely skilled attackers are breaking into internet networks. The new generation of Internet-enabled gadgets, which may be found everywhere in the globe, requires organizations to make plans, according to the research Internet of Things Wake-Up Call for Enterprises.
- Forbes study from 2015 states that a 42% rise in cloud-based security investment is anticipated. [2] Another study found that the investment on IT security climbed to 79.1% by 2015, indicating a rise of more than 10% annually. Security was identified as a top problem by 74.6% of commercial clients in 2011, according to International Data Corporation (IDC).[11] This publication compiles a number of peer-reviewed articles on cloud computing security concerns and mitigation techniques. Our research's goal is to comprehend cloud components, security concerns, and difficulties, as well as new technologies that may be able to reduce cloud vulnerabilities.
- Although it is widely acknowledged that the cloud has been a viable hosting platform since 2008, there is a notion that security in the cloud still has to be significantly improved in order to achieve higher rates of adoption at the business scale.[10] Many of the problems facing cloud computing, as noted by another study, need to be remedied right away. Although the industry has made tremendous strides in defending against threats to cloud computing, more work

has to be done before it reaches the level of maturity enjoyed by traditional/on-premise hosting.

- The author of this paper believes that corporate networks are distinct from the Internet at large and require particular attention because of preventive and detective attacks:[7] Security is of the utmost importance, centralized control is typical, and uniform, consistent procedures are crucial. However, offering robust protection is challenging and necessitates certain compromises. An open ecosystem with unrestricted communication and interoperability across all end hosts has definite advantages. However, it is also obvious that such openness leaves the network vulnerable to attacks from nefarious individuals both inside and outside of it.
- It is suggested that ENSASE be implemented in Evaluating and Enhancing Enterprise Network Safety Using Attacking Graphs. Large company network security assessments are difficult and time-consuming. We offer a new method that builds attack graphs that demonstrate how far inside and outside attackers may advance via a network by gradually compromising exposed and susceptible hosts by using configuration data on firewalls combined with vulnerability information on every network device.
- Within the context of Markov Decision Processes, optimal algorithms may be defined as constrained optimization problems where the goal is to maximize a certain utility subject to specified QoS restrictions. However, many poor algorithms have been developed since optimum algorithms are frequently computationally inefficient.
- In this part, we examine a variety of these specifically proposed methods for IEEE 802.16 networks. The lone exception is [12], where the authors suggested a contention-based TCP aware upstream routing for IEEE 802.16 connection. Most of these methods have been developed for real-time traffic with QoS guarantees.
- The BS analyses the send rate of every component flow dynamically and distributes resources depending on the observed send rate, whereas in, SSs do not submit any bandwidth queries for scheduling.[12] As the BS must keep account of the statuses for every TCP flows as well as the needs of the SSs, this type of dynamic transmit rate assessment of every TCP flows within the BS within each frame may result in scalability issues. Furthermore, the plan in does not take into

account the wireless channel's time-varying nature, the impact of RT T change on the need, or the impact of TCP timeouts.

- When resources are allocated only based on transmit rate, some flows may get starved, which will cause frequent congestion in the TCP window decreases and performance deterioration.[9] On the other hand, we provide a scheduling technique in this study that takes into consideration the time-varying wireless channel in addition to cwnd and TCP timeout.

2.2 Data Transfer Protocols , HTTP/ HTTPS

- Based on the injecting traffic and the control traffic, Chen et al.'s [1] analysis of TCP performance in multi hop wireless networks. The ideal congestion window size, where n is the total number of hops, determines infused traffic in large part. TCP's use is limited when a fixed cwnd is set. As a result, the literature concentrates more on decreasing control traffic to boost TCP performance.
- In [4]. examine how well the most popular VoIP codecs—G.711, G.723.1, and G.729—perform over a WiMAX network while using different service classes and NOAH as the transport protocol. With the capacity to transmit voice as packet over IP networks, Voice Over Internet Protocol is a potential new technology that enables voice communication via internet protocol-based networks. As a result, it can replace public switched telephone networks.
- The authors[2] suggest a receiver-aided technique in which the TCP receiver keeps track of the connection's contention state and, when necessary, notifies the TCP sender via the ACK mechanism. End-to-end latency is used by TCP receiver as a contention criterion.
- Focusing on the analysis of crucial QoS factors for Wimax Network [5] study. 500 mobile nodes in a WiMax network have had their throughput, latency, jitter, packet delivery ratio, and packet loss ratio (PLR) measured. The AODV protocol was chosen as a routing protocol due to its capacity to function well in erratic and highly mobile environments.
- Due to wireless router connectivity, WAP is "fast-paced technology" and simple to use since corporate operations are not constrained to a small space. In this manner, several devices may be accessed in the same location without

experiencing any connectivity issues. Costs are decreased because less cabling is required to connect each item individually [1].

- In contrast, sluggish WAP connections can occasionally cause user delays and other issues. Because the internet cannot be accessed, it is not completely secure, and the amount of data that can be collected is limited. WAP is pricey at first, even with a flawless keyboard and a good display [2].
- To support the findings of the literature research, this study collects various data, such as "An interoperability testing approach to wireless application protocols" and "micro-computed methodology". Several WAP strategies are emphasised by secondary research, including how it manages "High Latency," "Small Displays," "limited input facilities," and "Less Stable Connection" [4]. The WAP architecture was identified and the Wireless application protocol was compared to the Internet protocol using this technique.
- This study used secondary research approach to compile theoretical information regarding WAP [8]. This technique made it possible for this study to acquire sufficient data about the benefits and downsides of WAP in order to provide theoretical support for the literature review. The collected data will be subjected to thematic analysis. In this paper, the foundations of "Audio-Video testing" and "Cluster-based routing protocol in wireless sensor networks" are evaluated.

2.2 TCP Socket Programming

- The socket-based programming ideas introduced by Xue & Zhu (2009) are crucial for our investigation. They also gave a briefing on connection-oriented programming, which may ensure ongoing network connectivity.
- Understanding the fundamental terminology associated with client-server communication is made easier by insights on the principles, ideas, and designing aspect of client-server web applications.[17] Working on wireless connections is recommended to join the client-server connection. They also added that consumers will receive a wide range of services from these sorts of solutions.
- The communication strategy described here is applicable to socket programming and interface design. On File ServerJava Sockets are being used to construct a file server that is based on the TCP protocol. The server may react to requests for

files in a variety of formats. A dedicated Server keeps a list of files and is used for client authentication depending on the IP address of the connected client[13]

- The study of active networks has advanced steadily. It is important to build suitable execution platforms for active applications because active networks provide programmability to the Internet. Execution environments (EEs) are what these operating systems are known as, and a number of them have been developed, such as the Active Network Transport System (ANTS) and the Active Signalling Protocol (ASP) [12].
- A design framework for adding additional network protocols to the Internet was put out by Clark and Tennenhouse [1] in 1990. Active design of networks framework has gradually developed since the release of that position paper in the late 1990s. On the Internet, programme code and data may be transmitted concurrently thanks to the active network concept. Additionally, they could be performed and changed en route to their destinations.
- The peak performance of a certain kernel on various architectures is not guaranteed by OpenCL. Different programming techniques may be induced by the nature of the underlying hardware. The more common CPU architecture is undoubtedly the multi-core one. However, the upcoming Khronos proposal to bring GPU computation to the web is certain to increase programmers' interest in GPU architecture [15].
- Nevertheless, even if the most challenging aspect of parallel programming is hidden by OpenCL's API, a solid grasp of the underlying memory model results in more effective code. Details regarding the memory model are provided in the following paper, along with basic recommendations on how to construct an OpenCL cluster: [11].

CHAPTER 3 : SYSTEM DESIGN & DEVELOPMENT

The systems development cycle referred to as a development life cycle in system design, management of data, and software engineering, consists of creating, assessing, and executing the system. For this project's development, we utilized the following set of tools and development stack: Node.js and express for the application's hosting server, socket.io for web socket implementation, MongoDB for the database, and Docker for container creation.

3.1 Database Design and Architecture

Relationships are meaningful connections between tables that hold related data; they are what make databases functional. You might as well be dealing with fragmented spreadsheet documents instead of a database system if there is no connectivity among tables in a database.

3.1.1 One to One Relation

A record in one database may relate to only one value in another table (or, in certain situations, no records) in a one-to-one connection. One-to-one connections are uncommon since equivalent information may often be stored in the exact same table. Whether you break that information up into numerous tables is determined by your entire data structure and design technique; if you keep tables as tightly focused as feasible (as in a normalized database), one-to-one connections may be advantageous .

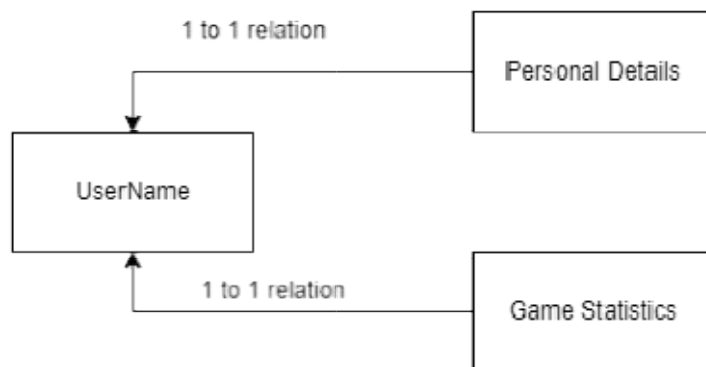


FIG 4. ONE TO ONE RELATION OF ITEMS IN DATABASE.

3.1.2 Database Design

I have employed MongoDB to design the database and its schema for the project . With replica sets, MongoDB provides high availability. A replica set consists of many copies of exactly the same data. Everyone who is part of the replica set is able to assume the role of a combination of the main copy or secondary replica at any moment. All data transactions are conducted on the original replica by default. Secondary replicas retain a copy of the primary's information via built-in replication. When an initial replica fails, the replica set performs an election that determines whichever secondary should take its place. Secondary data can be utilized to perform read operations if desired, however it is only eventually permanent by default. If the duplicated MongoDB deployment has only one primary member, an extra daemon known as an arbiter must be introduced to the set. It only has one responsibility: to settle the outcome of the next primary. As a result, even with a single primary and one secondary server, an idealized distributed MongoDB deployment needs at least three different servers.

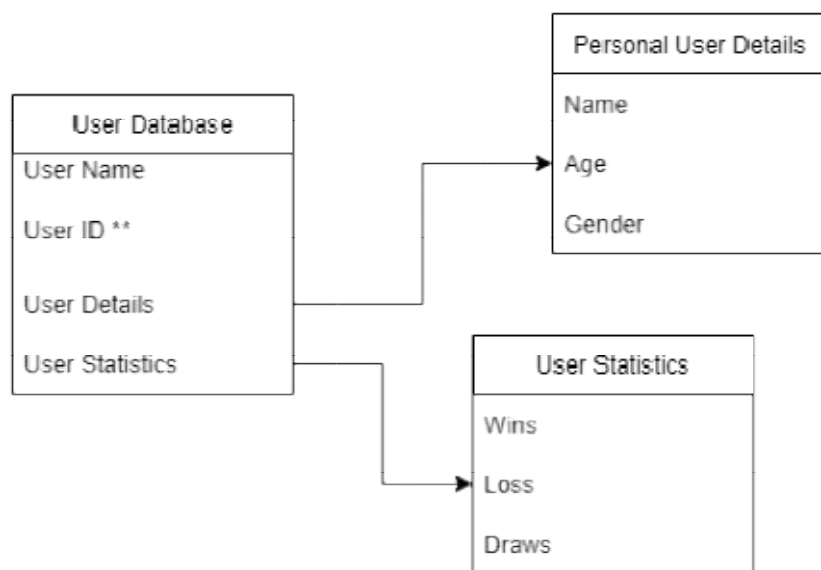


FIG 5. RELATION OF EACH VALUES AND USER DATABASE DESIGN.

3.2 System Design

Long-polling connections are used to communicate between the client computer and the server. Every message received by the server is sent out to all active connections. Web sockets are more dependable. Because an event may occur while some clients reconnect, a queue for messages should be carried out, and clients ought to transmit

the timestamp of the most recent message they received. This is not available in the present release.

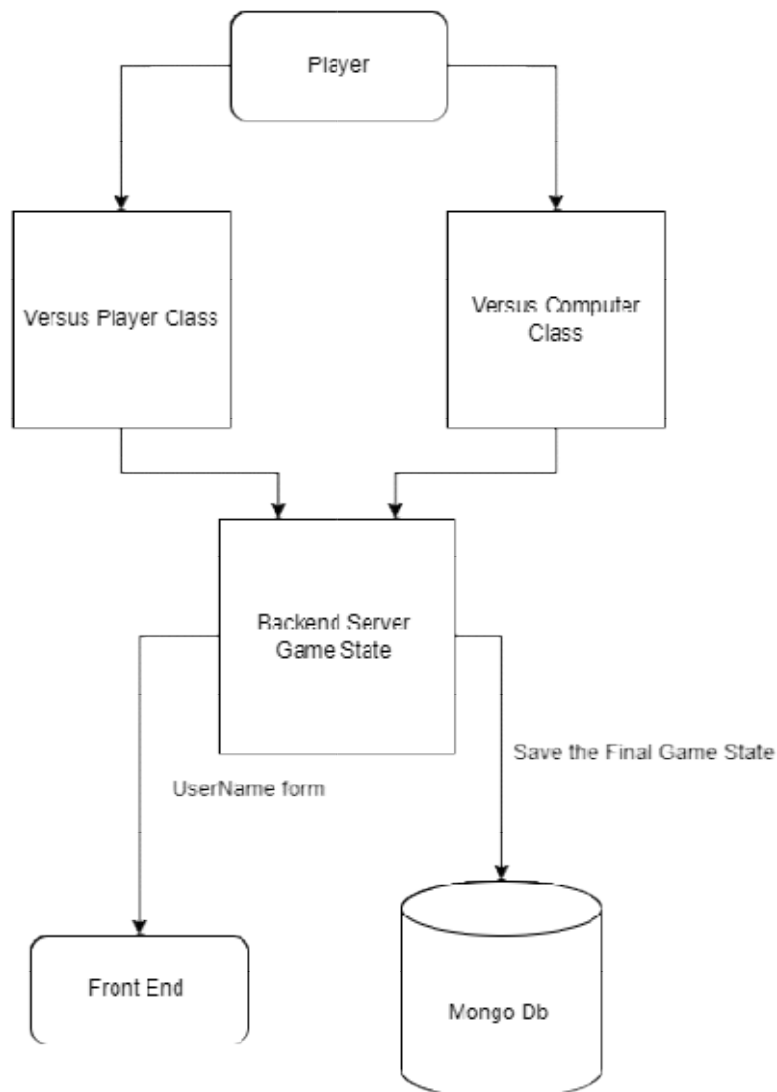


FIG 6. WORKFLOW OF SERVER AND DB

3.2.1 Web Server

The server answers to requests made via POST with json formatted data. Each message should include room and action parameters, as well as any additional parameters utilized by the related commands. The client side should send initialization as the initial command. While no reply is received, the player is the only one connected to the given room, and the game cannot be started. From the moment as there is a second player, the web server answers to the init call, informing each client of whose player it is.

When an individual moves, the set instruction is sent; the specified extra data is sent to every one of other connections observing this room. The wait command has no arguments; it simply adds the connection's information to the pool so that when someone sends set, the client receives the data.

The reset instruction (not implemented), which indicates that a fresh game between the players has begun and that all associated clients ought to reload their boards. Because the internet connection does not save the current state, any fresh watcher can observe only the latest moves and will be unable to initialize itself with the game's current state. The server, on the other hand, might save the state and, upon resumption, ask the first connecting players about the current game's state, while the client ought to issue a data command including all of the game data, that will be propagated to newly connected viewers or a rejoined player.

3.2.2 Client Side

The client is made up of three key parts: the Board, Game, and Connection classes. The connections class encapsulates the server connection and contains just one method, send, which has three parameters: the command that needs to be given out, the data to be sent with the command, and a callback to be invoked when there is a response. The answer is likewise a JSON thing, containing the command name as well as any data connected with it. If a connection fault occurs, the previous command will be resend with an increasing interval between attempts .

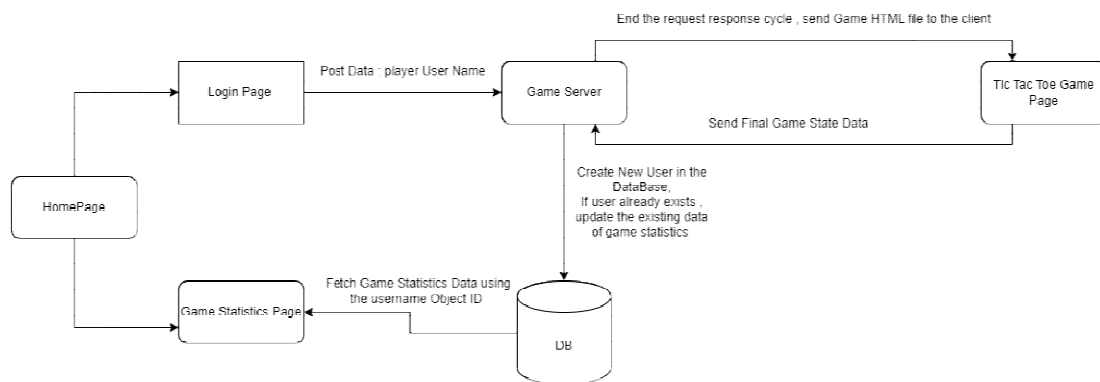


FIG 7.REQUEST RESPONSE CYCLE OF THE SYSTEM

The class initializes the link between the player and the board, as well as enabling and disabling the board based on whether or not it is the current user's time to move. The

Game class interacts with the board via event listeners and displays the status message. The board category is the game's heart. Its constructor takes an HTML Table component as the single argument and links the game boards to the table's cells. When the user clicks upon the cell's button, a board:set events are fired, which includes the row and number of the cell. If the move was successful, a deck:winning event is fired, and if the final square is filled but the move was unsuccessful, a deck:full event is fired.

3.3 ECMA-262

ECMAScript is a language for object-oriented programming used in a host environment to execute calculations and manipulate computational objects. ECMAScript as specified here is not meant to be computationally self-contained; in fact, there are no facilities in this specification for the entry of external data or the output of calculated results. However, it is expected that an ECMAScript program's computational environment will provide rather than only the objects along with the additional facilities characterized in this specification, but also certain context-specific host objects, the description of which and their behavior are outside the bounds of this specification with the exception to indicate that they might offer certain properties the fact that can be obtained and specific operations that may be invoked from an ECMAScript programme. A language for scripting is a type of computer language used to alter, modify, and automate a system's features. In such systems, beneficial functionality is already exposed to programme control through an operator interface, and the scripting language serves as a means of making it accessible to programme control. In this approach, it is claimed that the current system offers a host environment filled with objects and facilities that completes the scripting language's capabilities. Both seasoned programmers and novice programmers can utilize scripting languages. For client-side computing, a web browser offers an ECMAScript host environment, which may include objects that symbolize windows, menus, pop-ups, dialogue boxes, text regions, anchors, structures, history, cookies, and feedback/output. The host environment further offers a way to associate scripting code with activities like focus change, page and picture loading and unloading, failure and abort, selection, form delivery, and mouse movements. The HTML contains scripting code, and the page that is shown combines user interface components with

fixed and calculated text, pictures, and user interface elements. There is no requirement for a primary programme because the scripting code responds to user input.

3.4 Node

Node.js is an inexpensive and freely available server environment that runs on Linux, Windows, Unix, macOS, and other platforms. Node.js is a JavaScript runtime framework for the back end which processes code written in JavaScript outside of a web browser. The V8 Js Engine powers it. Node.js enables developers to create command-line tools, including JavaScript server-side scripts. The ability to run JavaScript script on the server that hosts the website is widely used to generate dynamic material for web pages before transmitting the page to the the customer's web browser. As a result, Node.js represents a "JavaScript every where" paradigm in which combining web application programming around a single language of programming rather than using several distinct languages for on the server side and client-side programming. Node.js' event-driven architecture enables for asynchronous I/O. Node.js allows the building of Web servers as well as networking tools by using JavaScript as well as a collection of components that handle several basic operations. Modules provide file system I/O, networking, binary data (buffers), cryptographic operations, data streams, and other vital services. Node.js modules use an API to make it easier to create server applications. Node.js only natively supports JavaScript, although there are other collate-to-JS languages available. As a result, Node.js applications may be written in a number of programming languages, such as Coffee Script, Dart, TypeScript, ClojureScript, and others.

Node.js is mostly used to create network programme like Web servers. The most notable distinction between Node.js and PHP is that most PHP functions block until conclusion, whereas Node.js operates are not blocking (commands perform simultaneously or additionally in parallel and use callbacks to indicate completion or failure).

3.5 Express

Express.js, or just Express, is a back-end web-based application framework for constructing Restful API using Node.js that is available for free and open-source use

under the MIT Licence. It is intended for the development of web-based applications and API. It has been dubbed the de-facto Node.js server framework defined as a Sinatra-inspired server, implying that it is very simple with numerous functionality accessible as plugins. Sharding allows MongoDB to scale horizontally. The user selects a shard essential which governs how information within a collection is dispersed.

Express is a node js web application framework. It provides a variety of capabilities for developing mobile and online apps, including the ability to create a single page, many pages, and a full hybrid web application. Express JS operates as a layer atop Node JS, assisting in the management of servers and routes. Express is a Node.js website development framework that offers a minimum and versatile collection of functionality for developing online and mobile apps. It speeds up the creation of Node-based Web applications. The factors that follow are some of the most important characteristics of the Express framework. Allows you to configure middleware to reply to HTTP requests. Defines a routing database for doing various actions according to the HTTP Method and URL. Allows for the dynamic rendering of HTML pages. The data is divided into ranges and spread among numerous shards depending on the shard key. (A shard consists of a master and one or more copies.) On the other hand, the shard key can be calculated to correspond to a shard, allowing for more even data distribution. MongoDB may be distributed over numerous servers, balancing the demand or replicating data in order to keep the system operational in the event of hardware failure.

3.4 Web Sockets

Web Sockets were created to provide full-duplex in communication among a client and a server, allowing data to move both directions instantly over a single open connection. The client's browser does not need to query an internet server for updates once the connection via Web Socket is established. Communication is instead bidirectional. When compared to HTTP/1's original long- and short-polling, this enhances speed and real-time capability. Web Socket does not adhere to any format. Web Sockets are popular because they allow you to transfer any type of data, including text and bytes. Some of this may seem familiar from the HTTP/2 portion, but it's crucial to remember that Web Sockets existed long before HTTP/2. We'll

compare them more thoroughly afterwards. TCP sockets are a basic notion in the TCP/IP application environment. The Transmission Control Protocol, also known as the TCP socket is a focused on connections socket that employs the TCP protocol. To establish a connection, three packets are required: the SYN package, the SYN-ACK package, and the acknowledgment package. The Internet Protocol (IP) address of the system and the port that it uses define a TCP socket. TCP sockets ensure that every data is obtained and recognized. TCP sockets are utilized to connect a server and an end-user process. The server code is executed first, which establishes a port and waits for incoming client connection requests. A user or server can send an electronic message after connecting to the exact same (server) port. When the communication is sent, it is processed by anyone who encounters it (server or client).

3.6 Modules

Modules in Node.js are isolated units of code that interface with a third-party app based on their relevant capabilities. Modules might be just one file or a group of files/folders. Because of their versatility and ability for breaking down an intricate piece of software into digestible bits, modules are frequently used by programmers.

3.6.1 Express Session

The HTTP protocol is the foundation of a website. HTTP is a protocol without states, which implies that both ends of the connection forget regarding one another at the end of each request and answer cycle. This is the point where the meeting comes into play. To maintain track of the user's status, a session will include some particular information about that client. Session-based authentication stores the user's state in the server's memory or a database.

When a client requests authentication, the server creates a session and stores it on the server. A cookie is sent by the server when it responds to the client. This cookie will hold the session's unique id, which was previously saved on the server and is going to remain on the client. The cookie in question will be delivered with each server request. To preserve an exact match among an ongoing session and a cookie, we utilize this session ID to search for the session maintained in the database itself or the session storage. HTTP protocol connections will now be stateful.

3.6.2 Cookie

Cookies is a Node.js module that allows you to get and set HTTP(S) cookies. Using cookies may be issued to prevent tampering. It may be used in conjunction with the node.js HTTP library or as Connect/Express middleware. The cookie is a combination of keys and values that the browser stores. Cookies are attached to each HTTP request submitted to the server by the browser. A cookie cannot hold a large amount of data. A session cookie is unable to save any user credentials or confidential information. If we did so, a hacker might simply obtain that information and take personal information for nefarious purposes. The session data, on the opposite, is saved on the server, in a file called a database or a temporary store. As a result, it can handle higher volumes of data. To retrieve data from the server, a session must be authenticated using a secret key or session id obtained from the cookie on each request.

3.6.3 Socket IO

Socket.IO is a driven-by-events library for real-time applications on the web. It facilitates real-time, bidirectional communication among web clients and servers. It is divided into two parts: a client side module that runs in a web browser and a server-side component for Node.js. Both components have nearly identical API. Socket.IO primarily use the Web Socket protocol, with polling as a backup option, while maintaining the same interface. Although it can be used just as a wrapper for Web Sockets it has many more features, such as transmitting to multiple sockets and storing data connected with each client, and asynchronously I/O. Within the Engine.IO threshold, a heartbeat system is built, enabling both the server's side and the client to detect when the other is no longer responding. Timers are established on both the server and the client, and expiration quantities (also known as the ping and timeout variables) are communicated during the connection handshake. Because of the timers, any future client calls must be routed to the same server, resulting in the sticky-session requirements when employing multiple nodes.

3.6.4 Bcrypt

Encryption aids in the integrity, security, and accessibility of systems and information. Our clients rely on our computerized systems to do business, offer services, and, ultimately, make a difference in the world by providing better and more affordable medicine. To offer a full comparison, Bcrypt has additional component that must be

discussed. A salt (or a random piece of data) can be added to a supplied password before it is passed through the hashing process. Salting enhances the difficulty of passwords as well as the length of time it takes to crack them using brute force. Rainbow Table assaults are similarly restricted. (Pap) Bcrypt was created for the purpose of password hashing and includes other components that boost the complexity of hashed passwords.

3.6.5 Flash

The flash storage area is a session-specific region used to store messages. Instructions are recorded to the flash and then cleared when the user sees them. Flash is commonly used in conjunction with redirects to ensure that the notification can be clicked to the next . After Express 3.x eliminated immediate support for the flash, this middleware was removed from Express 2.x. Connect-flash restores this feature to Express 3.x and any other middleware-compliant framework or application for extreme re usability.

3.6.6 Cookie Parser

Parse the Cookie header and add an object indexed by the cookie values to req.cookies. You may enable signature cookie support by giving a secret a string, which allocates req.secret so that other middleware can utilize it.

The middleware will scan the Cookie request on the request it receives and provide the cookie information as the req.cookies property and, if a secret was given, as the req.signed Cookies property. These are combinations of the cookies name and cookie value. When secret is supplied, this component will unsigned and check any signature cookie values before moving them from req.cookies to req.signed Cookies.

3.6.7 Mongo DB

MongoDB maintains records of data as documents (particularly BSON documents) organized into collections. A database contains one or more document collections. To specifically build a collection with different settings, such as selecting the maximum size or reviewing the validation criteria, MongoDB offers the db.createCollection() function. You don't have to explicitly construct the collection if you do not specify these settings since MongoDB generates new collections once you first save data for the collection. A collection of files does not need each document to have the identical schema by default; that is, the documents in one collection are not required to include

a comparable number of fields, and an information type for a field might vary among documents within a set of documents. MongoDB has two types of views: regular views and on-demand materialized views. Both of them view types return aggregation pipeline results.

```
const createGameSchema = async (client) => {
  await client.db(process.env.dbname).createCollection("gameUsers", {
    validator: {
      $jsonSchema: {
        bsonType: "object",
        title: "Tic Tac Toe Object Validation",
        required: ["username"],
        properties: {
          username: {
            bsonType: "string",
            description: "username must be string and is required",
          },
          email: {
            bsonType: "string",
          },
          password: {
            bsonType: "string",
          },
          Win: {
            bsonType: "int",
            minimum: 0,
            description: "Win must be integer",
          },
          Loss: {
            bsonType: "int",
            minimum: 0,
            description: "Loss must be integer",
          },
          Draw: {
            bsonType: "int",
            minimum: 0,
            description: "Draw must be integer",
          },
        },
      },
    },
  });
};
```

FIG 8. DATABASE VALIDATION SCHEMA.

Standard views are calculated when the view is read and are not saved to disc. Materialized views are saved and accessed from disc on demand. Standard views rely on the underlying collection's indexes. As a consequence, you cannot directly create, drop, or rebuild indexes on a conventional view, nor can you retrieve an array of

index on the view. On-demand materialized views outperform normal views in terms of read performance since they are accessed from disc rather than calculated as part of the enquiry. This speed advantage grows in proportion to the sophistication of the process and the amount of information being aggregated.

3.6.8 Net Module

The application model may be defined by combining many Compose files. The user-specified Compose file order **MUST** be followed when implementing a mix of YAML files by adding or overriding YAML components. Simple attributes, maps, and lists are merged by appending, whereas lists are overridden by the highest-order highest compose file. When merging complementary files that are housed in different directories, relative addresses **MUST** be resolved depending on the parent folder of the first Compose file. Combines **MUST** extend to the expanded version since some Compose file components can be written as either simple strings or complicated objects. For building flow-based TCP or IPC services (`net.createServer()`) or customers (`net.createConnection()`), use the `node:net` module's asynchronous network API. The local domain on Unix is also referred to as the Unix realm. The path is the name of a file system path. It is cut off at `sizeof(sockaddr_un.sun_path) - 1`, which is an OS-dependent length. On Linux and macOS, the typical amounts are 107 and 103 bytes, respectively. The Unix domains socket will also be unlinked when a Node.js API encapsulation generates one. For instance, `server.close()` will unlink a Unix domain socket that was created by `net.createServer()`.

3.7 MVC Design Pattern and Factory Design Pattern

Patterns of design are a means for you to organise the code in your solution so that you may profit from something, like quicker development times, code reuse, etc.

The object-oriented programming model lends itself to all patterns rather naturally. However, given the adaptability of JavaScript, you can also use these ideas in non-OOP projects. There are simply too many different design patterns to discuss them all in one article.

3.7.1 Model View Controller

MVC is only an architectural or design pattern applied to software engineering. While not a rigid rule, this pattern aids in helping developers narrow their attention to one particular aspect of their application at a time. The primary objective of MVC is to divide complex programme into distinct portions, each with a distinct function. Additionally, it enables secure logical application structuring, as we will demonstrate in this tutorial. Let's first examine what each feature of the pattern offers. A model is, as its name suggests, a design and structure. With MVC, the portion of the programme that communicates with the database is defined by the model, which also establishes how a database is organized. The attributes of a user which will be stored in our database will be defined here. Through the model, the controller has access to the database. Users interact with the programme through the view. In other words, this is the location of all the HTML template files. The controller communicates with the representation and provides the view with functionality and response. When an end user submits a request, the controller, which works with the database, receives it. Consider a controller as an attendant taking care of the orders of the patrons, in this instance the view. After then, the waiter, who represents the model/database or the controller processing the request.

The Model does not include any logic detailing how to show the data to a user; it merely holds the pure application data. (It is only data that is sent across the programme, for instance from the the front-end side to the database and from the back-end server view. The View shows the user the data from the model. The view is aware of how to get to the data in the model, but it is unaware of what this data signifies or how the user might be able to alter it. View just represents, shows the data from the programme on the screen. Among the view & the model lies the controller. The real company reasoning is written there. It keeps an eye out for events that are brought on by the viewpoint and responds to them appropriately. The response is often to invoke an operation on the model. The outcome of that action is then instantaneously reflected on the view since the views and the underlying model are linked by a notification system.

3.7.2 Factory Design Pattern

The factory pattern is one of the most common patterns for developing in Javascript. This type of design pattern is classified as a development pattern since it provides one of the best ways to create an item. We build items in the factory architecture without

exposing their origin logic to the client and utilise an interface that is universal to refer to freshly created objects. Simply provide an interface and an abstraction class to build an object using the factory architecture or factory technique pattern, and subsequently let the subclasses decide which class to instantiate. Subclasses, in a nutshell, are in charge of creating a class instance. The Factory Methods Pattern is a different acronym for the Virtual Constructor.

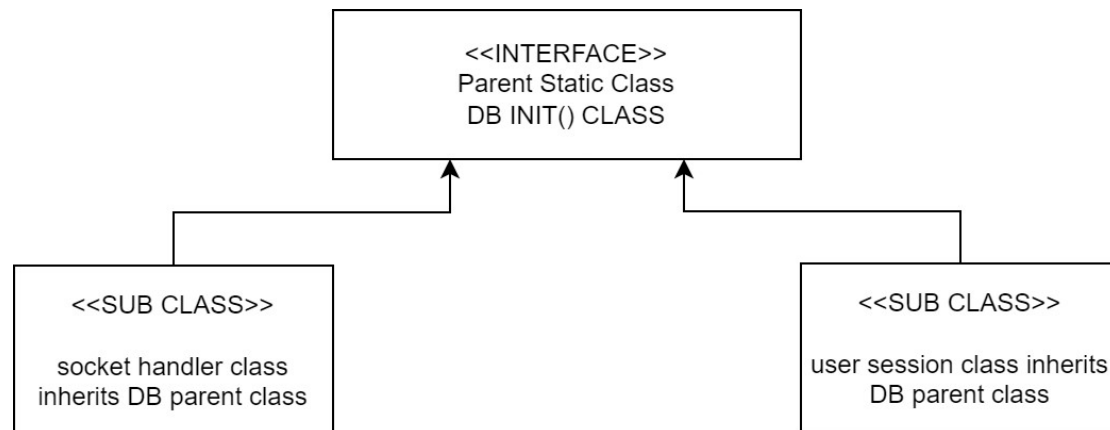


FIG 9. FACTORY DESIGN PATTERN

The factory pattern of design is used when a super class has numerous sub objects and we require returning one of the sub classes depending on the input. In this approach, the factory class takes responsibility of class instantiate rather than the client programme. Before delving into the advantages of the factory design arrangement .

3.8 Docker

Developers and system administrators may create, distribute, and execute programme using containers using the Docker platform. Containerization is the process of using containers to deliver software. Although using containers to quickly deploy applications is not new, it is. A container runs on Linux natively and shares the host computer's kernel with other containers. It is lightweight because it runs a separate process and uses no more memory than any other programme.

A virtual machine (VM), in contrast, utilizes a full-fledged "guest" operating system and has virtual access to host resources via a hypervisor. In general, VMs include a lot of overhead in addition to what your application logic uses.

A container is, to put it simply, a sand boxed process running on your computer that is separate from every other programme running on the host machine. This separation makes use of kernel namespaces and cgroups, two long-standing Linux features. These features have been made approachable and simple to utilize by Docker. The container itself is a functional instance of an image, to put it simply. Using the DockerAPI or CLI, you are able to start, stop, move, or delete a container. can be executed to the cloud or run on local or virtual machines. has portability. runs its particular software, binaries, and settings while being separated from other containers.It makes use of an isolated file system while operating a container. A container image provides this unique file system. The image must contain every dependency, configurations, scripts, binaries, and other materials required to run an application since it houses the container's file system. The image also includes metadata, variables related to the environment, a default command to run, and additional setup for the container.Decker's infrastructure as a service products deploy applications in containers using OS-level virtualization. Containers can communicate with one another via explicitly defined channels while remaining separate. In addition, they include their individual software, libraries in general, and configuration files.

3.8.1 Docker Compose

The version , facilities , the networks, volume, configure, and secrets are all defined in the Compose file, which is a YAML file. Compose files have a default location of either compose.yaml in the working directory or compose.yaml (recommended). For backward compatibility, Compose solutions Must incorporate docker-compose.yaml and docker-compose.yml. Compose implementations MUST choose the canonical compose.yaml file if both exist.The application model may be defined by combining many Compose files. The user-specified Compose file order MUST be followed when implementing a mix of YAML files by adding or overriding YAML components. Simple attributes, maps, and lists are merged by appending, whereas lists are overridden by the highest-order highest compose file. When merging complementary files that are housed in different directories, relative addresses MUST be resolved depending on the parent folder of the first Compose file.

Combines MUST extend to the expanded version since some Compose file components can be written as either simple strings or complicated objects.

One may design a container-based application that is platform independent using the Compose standard. An application of this type is created as a collection of containers that must operate cooperatively with sufficient shared resources and communication routes. A service is a computing component of an application. A service is a platform-based abstraction that is achieved by repeatedly executing the same container image (and settings). Through networks, services can communicate with one another. In this definition, a network is an abstraction of a platform feature used to provide an IP route between containers in related services. The Network specification contains low-level, platform-specific networking options, some of which maybe partly implemented. Persistent data is stored and shared by services in volumes. Such permanent data is described in the specification. A cloud platform's characteristics related to resource allocation on a cluster, replicated application distribution, and scalability are also available, along with other Windows container-specific attributes. We recognize that not every Compose implementation is expected to handle every characteristic and that platform-specific support for some properties can only be determined at runtime. The docker-compose tool, which is where the Compose file format was built, defined a versioned schema to control the permitted properties in a Compose file, however this description does not provide any assurance that the end-user characteristics will actually be implemented.

3.8.2 Docker File

There is no case difference in the instruction. To make it easier to identify them from arguments, it is customary to capitalize them. A Docker file is executed sequentially by Docker. A FROM directive must come first in a Docker file. This might come after comments, global scoped ARGs, and parser directives.

```
FROM node:lts-alpine
ENV NODE_ENV=production
WORKDIR /usr/src/app
COPY ["package.json", "package-lock.json*", "npm-shrinkwrap.json*", "./"]
RUN npm install --production --silent && mv node_modules ../
COPY . .
EXPOSE 3000
RUN chown -R node /usr/src/app
USER node
CMD ["node", "app.js"]
```

FIG 10.DOCKER FILE.

The original Parent Illustration that originates from you are constructing is specified by the FROM directive. Only one or more ARG guidelines, which provide parameters utilize by FROM sections in the Docker file, may come before a FROM instruction. Unless the line contains a valid parser directive, Docker interprets lines that start with a # as comments.

Anywhere else on a line with a #, an argument is considered. The manner that succeeding sections in a Docker file are treated depends on the presence of parser directives, which are optional. Parser instructions do not increase the build's layer count and are not displayed as an installation step. The format of parser directives is # directive = value, and they are expressed as a specific kind of comment. You may only use a directive once.

Docker stops looking for parser directives after processing a comment, an empty line, or a builder command. Instead of attempting to assess whether anything may be a parser directive, it considers everything structured as one as a comment. Therefore, the first line of every Docker file must have a parser directive.

3.8.2 Docker Compose File

A Docker application's services, the networks, and volumes are specified in the Compose file, which is a YAML file.

```
version: '3.4'

services:
  todonew:
    image: todonew
    build:
      context: .
      dockerfile: ./Dockerfile
    environment:
      NODE_ENV: production
    ports:
      - 3000:3000
```

FIG 11. DOCKER COMPOSE YAML FILE .

The Compose Specification outlines the most recent and suggested edition of the Compose file type. One may design a container-based application that is platform-independent using the Compose standard. An application of this type is created as a collection of containers that must operate cooperatively with sufficient shared assets and communication routes. A service is a computing component of an application.

```
version: '3.4'

services:
  todonew:
    image: todonew
    build:
      context: .
      dockerfile: ./Dockerfile
    environment:
      NODE_ENV: development
    ports:
      - 3000:3000
      - 9229:9229
    command: ["node", "--inspect=0.0.0.0:9229", "app.js"]
```

FIG 12. DOCKER COMPOSE-DEBUG YAML FILE .

A service is a platform-based abstraction that is achieved by repeatedly executing an identical container image (and settings). Through networks, services may communicate with one another. In this definition, a network is an abstraction of a platform feature used to provide an IP route between containers in related services. The Network specification contains low-level, platform-specific networking options, some of which MAY be partly implemented. Persistent data is stored and shared by services in volumes. The standard refers to such persistent data as a high-level, global file system mount. The Volumes specification contains actual platform-specific implementation information that MAY only be partly implemented on some systems. Some services call for configuration information that depends on the platform or runtime. Configure is a notion specifically defined in the standard to address this. Configure are similar to Volumes from the perspective of a Service container in that

they are mounted files. However, the real description calls for certain platform assets and amenities, which this type abstracts.

A particular type of config data for sensitive information that SHOULD NOT be revealed without security concerns is known as a Secret. Although platform-specific resources that offer sensitive data are unique enough to merit a separate concept and description inside the Compose standard, secrets are made accessible through services as directories mounted into their containers.

3.9 Socket Handler

The core logic is implemented using Socket IO module , the architecture of the TCP sockets can be explained using elaborate diagrams .

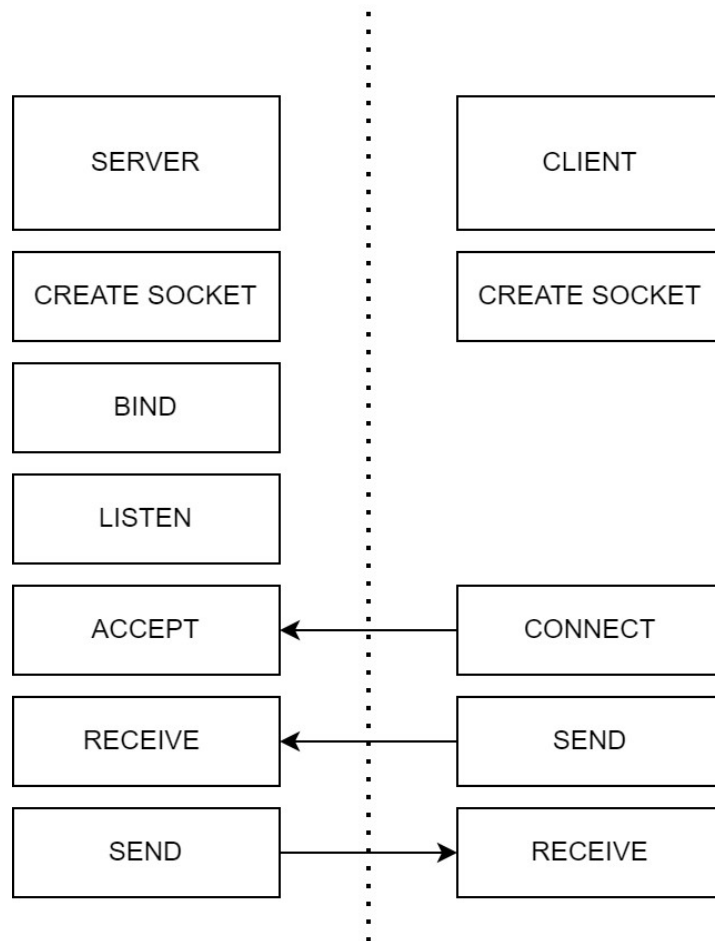


FIG 13. TCP SOCKET ARCHITECTURE

3.9.1 TCP Socket Data Transmission Model

When we look closely at how the number of ports are used in TCP/IP, it becomes clear how important the asymmetry among clients and servers is. Clients must be aware of the server process' port number since TCP and UDP are used to start application data transfers on the client side. In light of this, servers are obligated to utilize well-known port numbers. As a result, server processes are identified by well-known and recorded port numbers. They are utilized in requests submitted by clients as destination port numbers. Servers, on the other hand, react to users; they do not make contact with them. So, there is no requirement for the client to use a set aside port number. A server shouldn't utilize a well-known or listed port number to relay replies to clients, which is actually an understatement.

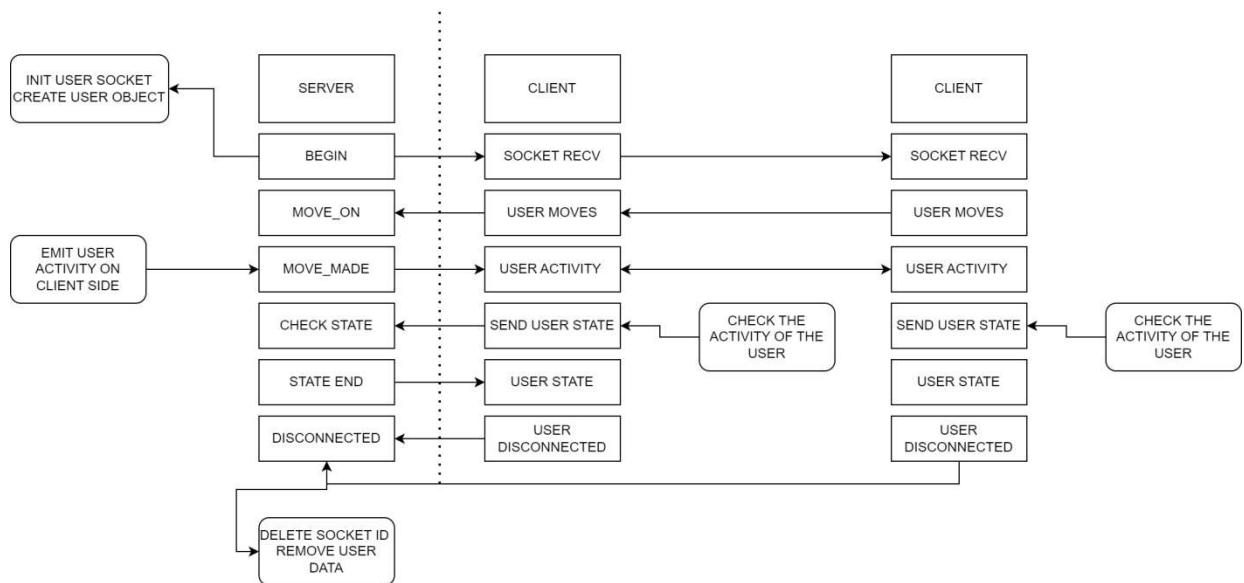


FIG 14. CLIENT - SERVER TCP SOCKET TRANSMISSION DESIGN

The explanation for this is that a given device may run the same protocol's client and server programme simultaneously. The HTTP server process on the client computer would receive the reply if a server accepted a request made via HTTP on port 80 of the client machine and transmitted the response to port 80 on the user machine rather than the HTTP client process that originated the request. The server has to be aware of the port number that the client is employing in order to know where to deliver the response. This is used by the server as the port used by the destination to transmit the response after being provided by the client's browser as the source port in the request. Client processes don't make advantage of popular or authorized ports.

Instead, an interim port number is given to each client process for use. A term used to describe this is ephemeral port number.

3.10 Socket Connection Algorithm

3.10.1 Server Side

- It uses the Socket to construct a brand-new game. Using the Socket, the createGame() function determines whether an opponent is already present. the function getOpponent().

```
static getOpponent(socket) {
  if (Socket.player[socket.id].opponent) {
    return Socket.player[Socket.player[socket.id].opponent].socket;
  }
  return false;
}
```

FIG 15. FUNCTION TO FETCH OPPONENT SOCKET ID .

- It transmits the "begin" event to both of the players utilizing the socket if an opponent is present. method emit(). The player's logo and turn are both sent in this event along with the initial game state data.

```
static createGame(socket) {
  if(Socket.opponent && socket.request.session.username === Socket.player[Socket.opponent].username){
    const delOppKey = Socket.opponent;
    delete Socket.player[delOppKey];
    Socket.opponent = null;
  }
  Socket.player[socket.id] = {
    symbol: "X",
    opponent: Socket.opponent,
    socket: socket,
    username: socket.request.session.username,
    turn: true,
  };
  if (Socket.opponent) {
    Socket.player[socket.id].symbol = "O";
    Socket.player[socket.id].turn = false;
    Socket.player[Socket.opponent].opponent = socket.id;
    Socket.opponent = null;
  } else {
    Socket.opponent = socket.id;
  }
}
```

FIG 16. FUNCTION TO CREATE SOCKET OBJECT.

- It watches for the player who is now in control of the turn to emit the "move" event. The "move_made" event is sent to both players over the socket whenever a move is made.
- Updating the game state.method emit(). The most recent game state and turn information is sent via this event.

```

socket.on(socket_options.move, (player_postition) => {
  if (Socket.player[socket.id].turn) {
    Socket.player[socket.id].turn = false;
    Socket.player[Socket.getOpponent(socket).id].turn = true;
    socket.emit(socket_options.move_made, {
      position: player_postition,
      turn: Socket.player[socket.id].turn,
      symbol: Socket.player[socket.id].symbol,
    });
    Socket.getOpponent(socket).emit(socket_options.move_made, {
      position: player_postition,
      turn: Socket.player[Socket.getOpponent(socket).id].turn,
      symbol: Socket.player[socket.id].symbol,
    });
  }
});

```

FIG 17. SOCKET FOR USER STATE .

- It monitors either player's "state_check" event to determine the game's current state. It emits the "state" event with the "Win" flag set to true and the "Loss" flag

```

if (Socket.isWinner(gameBoard)) {
  socket.emit(socket_options.state, {
    Win: true,
    Loss: false,
    Draw: false,
    turn: Socket.player[socket.id].turn,
  });
  Socket.getOpponent(socket).emit(socket_options.state, {
    Win: false,
    Loss: true,
    Draw: false,
    turn: Socket.player[Socket.getOpponent(socket).id].turn,
  });
} else if (Socket.isDraw(gameBoard)) {
  socket.emit(socket_options.state, {
    Win: false,
    Loss: false,
    Draw: true,
    turn: Socket.player[socket.id].turn,
  });
}

```

FIG 18. SOCKET FOR STATUS CHECK .

- set to false if there is a winner and the opposite if there is a loser. It emits the "state" event with the "Draw" flag set to true if the game is a tie.
- The user score is updated in the database using the db.updateUserScore() function as soon as a player emits a "end" event.
- In order to manage the situation where a player disconnects from the game, it also listens for the "disconnect" event.

```
static isWinner(gameBoard) {
  const matches = ["XXX", "000"];
  const winCombination = [
    gameBoard.cell11 + gameBoard.cell12 + gameBoard.cell13,
    gameBoard.cell14 + gameBoard.cell15 + gameBoard.cell16,
    gameBoard.cell17 + gameBoard.cell18 + gameBoard.cell19,
    gameBoard.cell11 + gameBoard.cell15 + gameBoard.cell19,
    gameBoard.cell13 + gameBoard.cell15 + gameBoard.cell17,
    gameBoard.cell11 + gameBoard.cell14 + gameBoard.cell17,
    gameBoard.cell12 + gameBoard.cell15 + gameBoard.cell18,
    gameBoard.cell13 + gameBoard.cell16 + gameBoard.cell19,
  ];
  for (const element of winCombination) {
    if (element === matches[0] || element === matches[1]) {
      return true;
    }
  }
  return false;
}
```

FIG 19. SOCKET FOR STATE OF BOARD .

- When a player disconnects, the other player receives the "opponent_left" event via the Socket.getOpponent() function. The surviving player is informed by this occurrence that their opponent has departed the game.

```
socket.on(socket_options.end, (gameState) => {
  db.updateUserScore(Socket.player[socket.id], gameState);
});
socket.on(socket_options.disconnect, () => {
  if (Socket.getOpponent(socket)) {
    Socket.getOpponent(socket).emit(socket_options.opponent_left);
  }
});
```

FIG 20. SOCKET FOR DISCONNECTION.

3.10.1 Client Side

- The server records the user's ID to the console whenever a client joins to it.
- The server provides player data to the client when the game starts.
- The server refreshes the game grid and verifies the game state after receiving move data from the client when a player performs a move.
- When the server gets a game status check, it evaluates whether the game should conclude in a winner, a draw, or a new round, and then delivers the relevant game state information to the clients.
- If the opponent leaves the game, the server sends a message to the client indicating that the opponent has left the game.

CHAPTER 4 : EXPERIMENTS & RESULT ANALYSIS

By building a test bench to evaluate the selected functionality while monitoring the necessary metrics, such as throughput or memory use, performance checks and comparison are frequently made. I needed to learn how to measure the performance of web servers in order to conduct a study that would enable me to find an answer to my research issue. What other people have done and what criteria are important. I completed this through reading scholarly publications and researching other people's work. When examining the efficiency and burden of various frameworks, protocols, and network services, it has been demonstrated that some metrics are common to assess and some queries are common to answer. How many inquiries or messages may be sent or received per second at varying levels of concurrency, data transmission rates, and response times

The Benchmark / criteria that were considered to measure the performance of socket .io module with other TCP socket modules such as Socket Js , web sockets , net module.

- With various client levels, different levels require varied amounts of time to connect.
- With various tiers of customers, the time it takes for them to get a message (after a connection has been made) varies.
- With varying client tiers, the server's memory use varies.

The server answers to the clients' connection requests, and a connection is formed. Following that, the server notifies the client, and assessments are made of both the server & the clients. The client cuts off communication after receiving the message.

To as closely as possible replicate the actual application, the messages are transmitted in exactly the same length & format as those that are now being sent . The message's text is a JSON string containing key:value pairs that include, but are not limited to, the thunderstorm's timestamp, location, and proximity.

4.1 Experiment Parameters

The pre-exist questions are examined and measured in several methods that are described below.

4.1.1 Parameter I : Time to establish a link between client and web server

The built-in function `process` in the Node.js framework is used to calculate the answer to the query "How does the time needed to establish a relationship change with various numbers of active clients?" With nanosecond precision, `hrtime()` returns the current excellent quality real time.

The time that the function call returns is a relative time that is resistant to clock movement and suitable for bench-marking .

4.1.2 Parameter II : Time to receive an acknowledgement during a link up

How does the amount of active clients affect the time it takes to receive an alert (after a connection has been made)? is built up according to subsection 3.3.1. The first epoch is obtained when the connection is made, which is the difference. After it obtains the communication from the server, a second timestamp is obtained. Then, certain statistics are computed and reported, including the lowest, maximum, and median times in addition to the standard deviation of the times.

4.1.3 Parameter III : Memory consumption on web server

In the Node.js setting, the built-in function `process` is used to measure the memory utilization on the server as a function of the number of active clients. `memory Usage()` provides extensive information on how much memory the process is presently using. For instance, the V8 engine's memory consumption as well as the Resident Set Size (henceforth termed RSS). The RSS measures how much room a process takes up in the main memory. containing all JavaScript code and objects. Since RSS represents the actual amount of memory needed to operate the operation, it is data in its entirety that is utilized for monitoring. The maximum RSS currently in use is only written out once per second but is refreshed ten times each second.

4.2 Result Analysis

Results and information related to the questions presented in subsection 4.1 are provided in this section.

4.2.1 Parameter I : Time to establish a link between client and web server

Plain Web Sockets were able to create a connection with just 1 client in 8 milliseconds. Socket.IO was around three times slower than net module, and vice versa. Up to 1000 clients, plain Web Sockets continue to be around 3 times quicker than Socket.IO or net module. Plain Web Sockets took 211 milliseconds with 10,000 concurrent clients, whereas Socket.IO took 319 milliseconds. net module failed to keep up but reached its maximum of 7000 users at 470 ms.

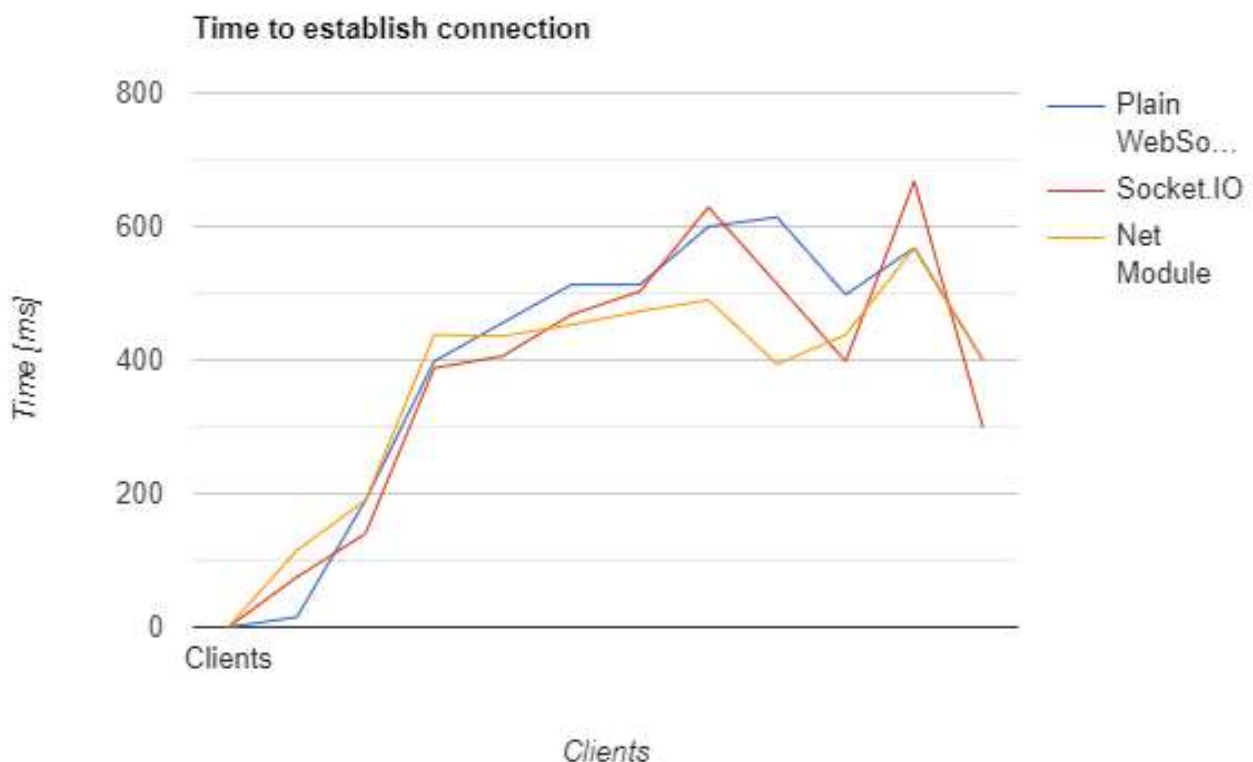


FIG 21. GRAPH FOR PARAMETER I

4.2.2 Parameter II : Time to receive an acknowledgement during a link up

Plain Web Sockets took just 1 millisecond (ms) with just 1 client to get a response from the server, compared to 21 ms and 3 ms for Socket.IO and net module, respectively. Plain Web Sockets took 63 milliseconds with 1000 simultaneous clients, but Socket.IO took 425 milliseconds and net module took 47 milliseconds. With 10000 clients, regular Web Sockets took 366 milliseconds while Socket.IO took 483 milliseconds. net module reached its maximum client count at 481 milliseconds.

The network footprints for 1 client delivering 1 message were captured and stored with the network's trace analyzer Wire shark in order to better understand how and how the architecture may vary. The quantity of data required to send a message varies. Reassembled TCP Sections for plain Web Sockets total 75963 bytes. While it is 96262 bytes for Socket.IO and 96012 bytes for net module. The level of overhead in the platforms is also shown by the number of transmissions; standard Web Socket utilizes 5, but Socket.IO and net module each require 13 and 8 transmissions, respectively.

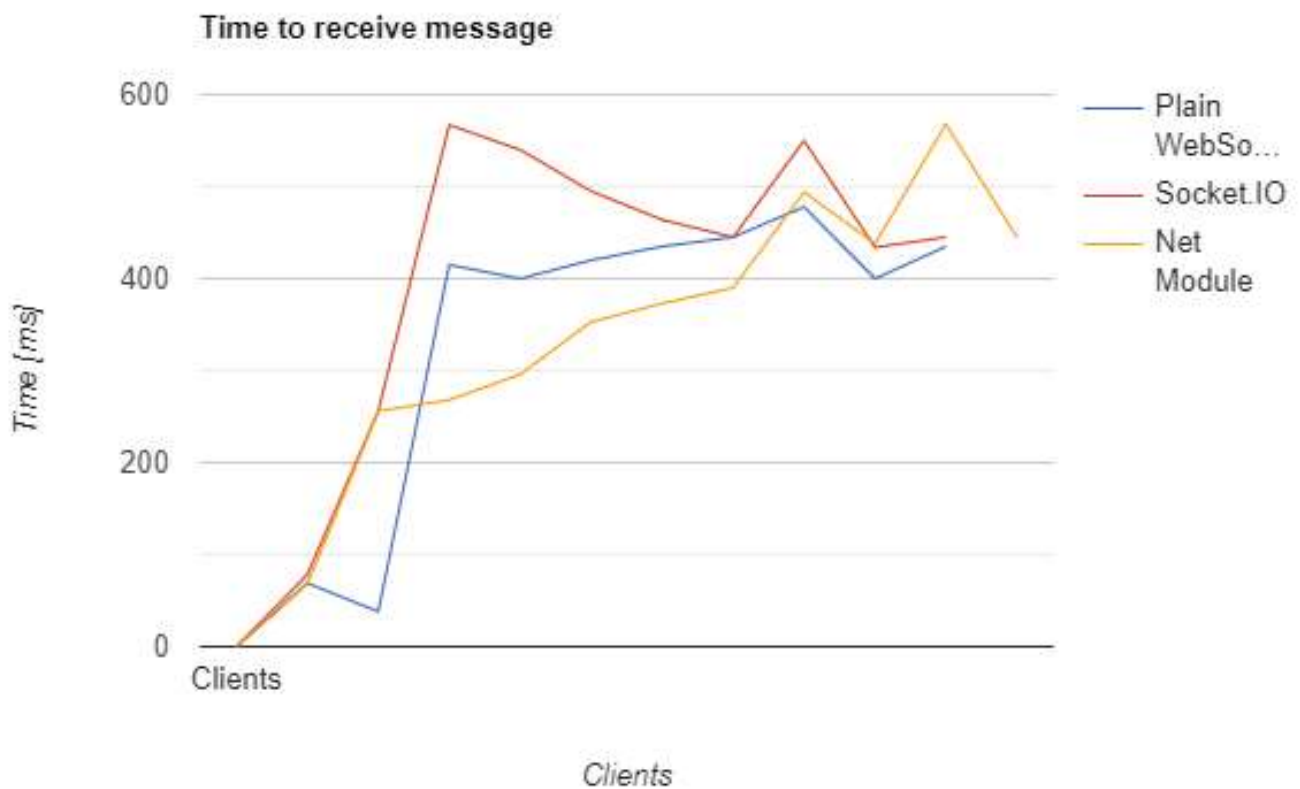


FIG 22. GRAPH FOR PARAMETER II

4.2.3 Parameter III : Memory consumption on web server

No matter the framework, the server used about 20 MB of RAM with just 1 client. Simple Web Sockets used about 76 MB with 1000 clients, whereas Socket.IO needed about 200 MB. 84 MB were needed for net module. Plain Web Sockets needed 210 MB of RAM with 10,000 clients, whereas Socket.IO needed over 2 GB. The maximum number of clients for net module was 7000, needing 341 MB of RAM.

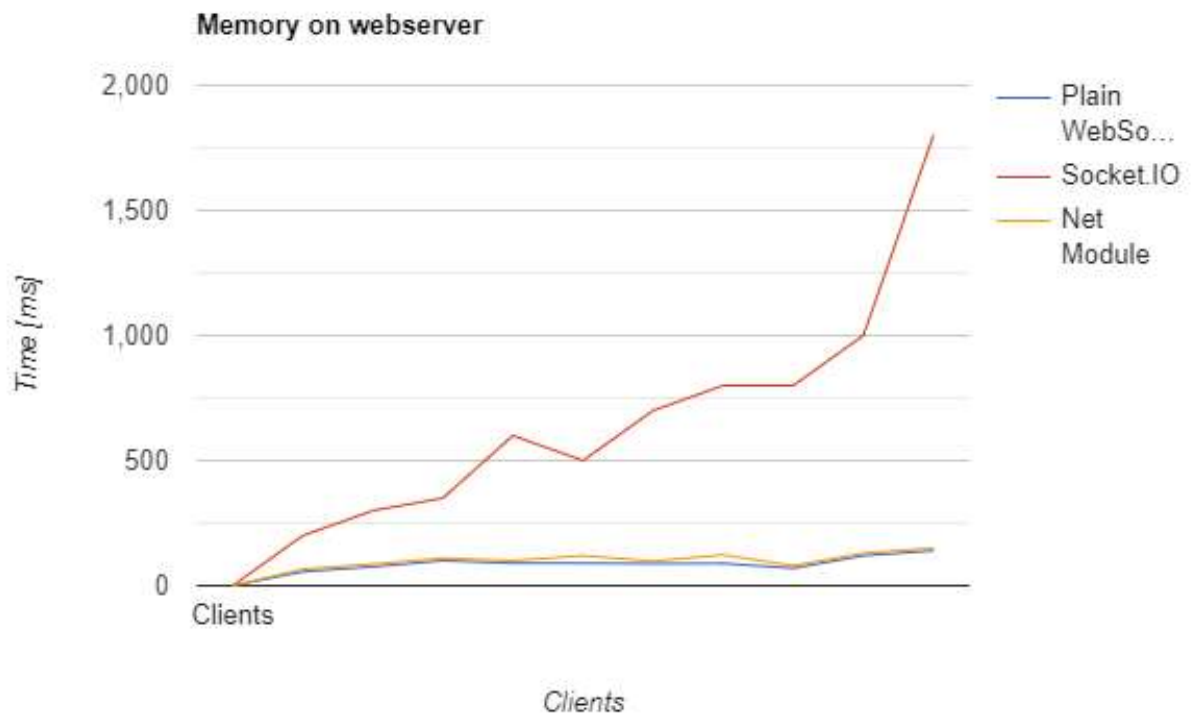


FIG 23. GRAPH FOR PARAMETER III

CHAPTER 5 : CONCLUSIONS

As anticipated, simple Web Sockets are up to three times quicker than the frameworks like Socket.IO and net module. This is anticipated since Socket.IO and net module contain additional cost in the form of transmitting several data packets, but simple Web Socket follows the protocol specification Messages and requests are only sent in order to establish a connection between devices. When establishing a connection, Socket.IO sends three requests for data over The Hypertext Transfer Protocol and two messages via Web Socket. Prior to attempting to build a Web Socket connection that is confirmed, it first opens an XHR polling connection. The overhead is perceived as requiring more transmissions in net module initially makes a standard HTTP GET request, and the server replies with a JSON object . Following that, information about the URL and server is provided along with the Web Socket request responds. Finally, the server sends a message to the client. There are considerable speed advantages to be had when utilizing plain Web Sockets with up to 2000 simultaneous clients. All frameworks are nearly equivalent when there have 3,000 or more customers. This, in my opinion, was caused by the gigabit network to which the test bench was attached. It appears that while the server is already sending information to the connected clients, the gigabit Ethernet prevents the server & clients from connecting any quicker.

Two key advantages may be attained by selecting a framework with the right level of abstraction. less time and money spent on development. Additionally, apps that have been optimized are quicker, more stable, and use less power. Being able to handle the amount of work with less resources, like one server, is another benefit. As opposed to needing several servers to handle the odd significant surge in user traffic, while doing so most of the time with an idle workload that has been demonstrated to be quite inefficient and still use a lot of power,

REFERENCES

- [1] C. DiCesare and J. A. Hoxmeier. An experimental study of browser-based applications was published in "System Response Time and User Satisfaction." In: Association of Information Systems Americas Conference Proceedings , Jan. 2000.
- [2] K. Ross and J. Kurose. 7th edition of Computer Networking: A Top-Down Approach , 2017 .
- [3] A. Melnikov and I. Fette. The Web Socket Protocol , 2011.
- [4] Performance Comparison and Evaluation of Web Socket Frameworks: Netty, Undertow, Vert.x, Grizzly, and Jetty, Y. Wang, L. Huang, X. Liu, T. Sun, and K. Lei. In: Hot Information-Centric Networking, the first IEEE International Conference , 2018 .
- [5] M. Horvat, S. Srbljic, and D. Skvorc. "Performance Evaluation of the Web socket Protocol for the Implementation of Full-Duplex Web Streams." Submitted to the ,2014 .
- [6] Steven K. Reinhardt, Erik G. Hallnor, and Nathan L. Binkert M5-based network-oriented full-system simulation. Using Commercial Workloads in the Sixth Workshop on Computer Architecture Evaluation , 2013 .
- [7] At the Forge: Communication in HTML5 by R. M. Lerner, Linux Journal, 2011.
- [8] "Cross-Layer-Based Modelling for Quality of Service Guarantees in Mobile Wireless Networks," IEEE Communications Magazine, X. Zhang, J. Tang, H.-H. Chen, S. Ci, and M. Guizani , 2006 .
- [9] R. Ludwig and R. H. Katz, "The Eifel algorithm: Making TCP robust-against Spurious Re-transmissions," ACM SIGCOMM Computer Communication Review, 2000.

- [10] S. Phoemphon, C. So-In and N. Leelathakul, "Fuzzy weighted centroid localization with virtual node approximation in wireless sensor networks", *IEEE Internet Things J.*, 2018.
- [11] A. A. Sorokin, V. N. Dmitriev and N. N. Losev, "Virtual laboratory for modeling and studying of telecommunication systems based on software package network simulator", *Vestnik Astrakhan State Tech. Univ. Ser. Manage. Comput. Sci. Inform.*, vol. 1, no. 2010, pp. 103-108, 2010.
- [12] J. Samain, G. Carofiglio, L. Muscariello, M. Papalini, M. Sardara, M. Tortelli, et al., "Dynamic adaptive video streaming: Towards a systematic comparison of ICN and TCP/IP", *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2166-2181, Oct. 2017.
- [13] J. Bai, W. Wang, M. Lu, H. Wang and J. Wang, "TD-WS: A threat detection tool of Web Socket and Web storage in HTML5 websites", *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5432-5443, 2016.
- [14] J.-S. Wang and G.-H. Yang, "Data-driven methods for stealthy attacks on TCP/IP-based networked control systems equipped with attack detectors", *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 3020-3031, Aug. 2019.
- [15] L. Quan, Z. Xu and Z. Li, "Implementation of hardware TCP/IP stack for DAQ systems with flexible data channel", *Electron. Lett.*, vol. 53, no. 8, pp. 530-532, 2017.
- [16] H. K. Rath and A. Karandikar, "On TCP-Aware Uplink Scheduling in IEEE 802.16 Networks," in *Proc. of IEEE COMSWARE*, January 2008.
- [17] H. K. Rath, A. Bhorkar, and V. Sharma, "An Opportunistic Uplink Scheduling Scheme to Achieve Bandwidth Fairness and Delay for Multicast Traffic in Wi-Max (IEEE 802.16) Broadband Wireless Networks," in *Proc. of IEEE GLOBECOM*, November 2006 .

- [18] A. Gomez-Sacristan, V. M. Sempere-Paya and M. A. Rodriguez-Hernandez, "Virtual laboratory for QoS study in next-generation networks with metro Ethernet access", *IEEE Trans. Educ.*, vol. 59, no. 3, pp. 187-193, Aug. 2016.
- [19] Lin Zhou, *Computer Network Engineering [M]*, People's Posts and Telecommunications Press, 2013.
- [20] Chen Yingming, *Computer Network and Application [M]*, Metallurgical Industry Press, 2011.