International Journal of Scientific Research in Computer Science and Engineering
Vol.**12**, Issue.**4**, pp.**66-76**, August **2024**
E-ISSN: 2320-7639
Available online at: www.isroset.org

Research Article

Internet Protocol with Internet Programming (IP with IP): Architecture and Design

Arun Kumar Singh[1*], Alak Kumar Patra[2]

[1]School of MCS, PNG University of Technology, Lae, Papua New Guinea
[2]School of Civil Engineering, PNG University of Technology, Lae, Papua New Guinea

**Corresponding Author: arunsinghiiita@gmail.com*

Received: 23/Jun/2024; **Accepted**: 25/Jul/2024; **Published**: 31/Aug/2024

Abstract— The Internet Protocol (IP) is a communication protocol utilized for inter-device communication on the Internet. It is a protocol at the network layer that furnishes an addressing system for uniquely identifying devices on the network and facilitating the routing of data packets between them. Conversely, Internet Programming pertains to the creation of applications and services that operate on the Internet. This encompasses activities such as web development, mobile app development, and other types of networked application development. IP is a crucial element of Internet programming because it establishes the underlying infrastructure for inter-device communication on the Internet. Internet programmers must possess a comprehensive understanding of IP and its associated protocols, like TCP (Transmission Control Protocol) and UDP (User Datagram Protocol), to construct applications that can communicate over the Internet. Additionally, internet programmers employ a variety of programming languages, frameworks, and tools to build applications that function on the Internet. This can incorporate languages such as JavaScript, Python, Ruby, and PHP, as well as web development frameworks like React, Angular, and Vue. In summary, IP and Internet programming are closely connected, as IP forms the basis for inter-device communication on the Internet, while Internet programming utilizes this foundation to develop a wide range of networked applications and services. In this article, we will elucidate the significance of both IP and internet programming as indispensable constituents of the Internet, demonstrating their close interconnection. IP establishes the groundwork for data transmission, while internet programming furnishes the means for creating and delivering web-based applications and services.

Keywords— Internet Protocol (IP); Internet Programming; IPv4; IPv6; Type of Service; Protocol

1. Introduction

The Internet Protocol (IP) is a communication protocol utilized for transmitting data over the internet. It facilitates the transfer of data packets between internet-connected devices. At the network layer of the OSI model, IP operates to transmit data between networks. Conversely, internet programming pertains to the creation of internet-accessible applications and software. It entails developing web-based applications that users can access through a web browser or specialized application. Internet programming commonly employs several protocols, including IP, to enable communication between devices and servers on

the internet. Developers utilize programming languages like HTML, CSS, JavaScript, and others to construct web-based applications [1].
The utilization of IP in internet programming is critical as it enables devices to communicate with servers on the internet. This communication is vital for the operation of web-based applications and services, granting users the ability to access information and interact with various systems and services online. Overall, IP and internet programming are closely intertwined, with IP serving as a fundamental protocol for internet communication and internet programming utilizing diverse protocols to create web-based applications capable of functioning on the internet. On the other hand, internet programming involves the practice of coding to develop web-based applications or services, utilizing programming languages and tools to construct software accessible through a web browser [2].

- IP is indispensable for internet programming since it establishes the basis for transmitting data over the internet. When a user accesses a web-based application, the IP protocol is employed to send data from the application server to the user's computer.
- Internet programming encompasses an array of languages and technologies, such as HTML, CSS, JavaScript, as well as various server-side programming languages like PHP,

Python, and Ruby. These programming languages are employed to create the user interface, functionality, and database connections for web-based applications.

- The Internet Protocol (IP) is a fundamental protocol employed for internet communication. It is responsible for routing data packets between internet-connected devices. IP operates at the network layer of the OSI model, which facilitates data transmission between networks.
- IP provides a standardized approach for data transmission between devices on the internet. It utilizes unique IP addresses to identify internet-connected devices, and data packets are directed from one device to another based on these addresses. Additionally, IP defines the structure of data packets, including the header, which contains information like the source and destination IP addresses, and the payload, which comprises the actual transmitted data.
- IP is a connectionless protocol, which means that data packets can be sent between devices without establishing a dedicated connection between them. This makes IP a highly scalable protocol, as it can handle large amounts of data traffic across multiple devices and networks.
- IP is used in conjunction with other protocols, such as TCP (Transmission Control Protocol) and UDP (User Datagram Protocol), to provide reliable and efficient communication over the internet. TCP provides reliable, ordered, and error-checked delivery of data packets, while UDP provides a faster, connectionless delivery of data packets with no guarantee of delivery.
- Internet Protocol (IP) is a set of rules and procedures that govern the communication of data packets between devices on the internet. IP operates at the network layer of the OSI (Open Systems Interconnection) model and is responsible for routing data packets from one network to another.

When a device sends data over the internet, the data is divided into small units called packets. Each packet contains a header that includes information such as the source and destination IP addresses, the packet sequence number, and the type of protocol used. The packet also contains the actual data being transmitted. IP uses a unique IP address to identify each device connected to the internet. An IP address is a 32-bit number (IPv4) or a 128-bit number (IPv6) that uniquely identifies a device on the internet. IP also supports various protocols for data transmission, including TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). TCP provides reliable data transmission by ensuring that packets are received in the correct order and retransmitting any lost packets. UDP, on the other hand, provides faster data transmission by not performing error checking or retransmitting lost packets. IP operates at the network layer of the OSI model, which is responsible for the transmission of data between networks. It is a connectionless protocol, meaning that it does not establish a dedicated end-to-end connection before transmitting data. IP provides two main functions: addressing and routing. IP addresses are used to identify devices connected to the internet, and they consist of a series of numbers separated by dots (e.g., 192.168.0.1). IP routing involves the forwarding of data packets between devices connected to the internet, based on the destination IP address of the packet [3].

IP has two main versions in use today: IPv4 and IPv6. IPv4 uses 32-bit addresses and is still the most widely used version of IP, but it is limited in the number of unique addresses it can support. IPv6, on the other hand, uses 128-bit addresses and provides a much larger address space, which is necessary to support the growing number of devices connected to the internet. IP provides a unique identification number, called an IP address, to each device connected to the internet. This IP address allows devices to communicate with each other over the internet. When a device wants to send data to another device, it breaks the data into small packets and adds the IP address of the destination device to each packet. The packets are then sent through various routers on the internet, which use the IP addresses to forward the packets towards the destination device. Once the packets reach the destination device, they are reassembled into the original data. In addition to increased address space, IPv6 includes other improvements over IPv4, including improved packet handling, simplified network configuration, and enhanced security features. Despite the benefits of IPv6, the transition from IPv4 to IPv6 has been slow due to various factors, including the complexity of implementing IPv6, the need for backward compatibility with IPv4, and the cost of upgrading network infrastructure. IPv6 also includes additional features such as better security, better support for mobile networks, and simplified packet processing. However, IPv4 is still widely used and will continue to be used for many years to come. One of the main differences between IPv4 and IPv6 is the format of the IP address. IPv4 addresses are written in decimal format, separated by periods, while IPv6 addresses are written in hexadecimal format, separated by colons. For example, an IPv4 address may look like 192.168.1.1, while an IPv6 address may look like 2001:0db8:85a3:0000:0000:8a2e:0370:7334 [4].

2. IP Version: IPv4 and IPv6 with Internet Programming

Internet programming involves the development of web applications and services that can be accessed online. The field heavily relies on both IPv4 and IPv6, which are essential protocols for internet communication. IPv4 has long been the predominant protocol used for internet communication, and most applications and services are designed with IPv4 addresses in mind. As a result, internet programmers must ensure that their applications are compatible with IPv4 addresses. Internet programming encompasses the creation of software and applications that can be utilized over the internet. The two versions of the Internet Protocol, IPv4 and IPv6, have a significant influence on internet programming, particularly in terms of network communication. Communication protocols, including IP, play a vital role in facilitating data transmission between online devices in internet programming. The choice between IPv4 and IPv6 can affect the programming of these applications. Developers may need to consider factors such as the maximum size of supported data packets for each IP version, the addressing system in use, and the available security features. IPv4 has been extensively utilized in internet programming, and the majority of internet applications are designed to work with

IPv4 addresses. However, the limitations of IPv4 have become evident due to the increasing demand for internet connectivity and the growing number of connected devices. This has resulted in the adoption of IPv6, which provides a larger address space and additional features, including built-in security. To ensure that internet applications can reach the maximum number of users and take advantage of the new features offered by IPv6, developers must ensure compatibility with both IPv4 and IPv6. This involves testing the application's compatibility with both IP versions and ensuring its functionality with both IPv4 and IPv6 addresses. In conclusion, network communication in internet programming is significantly impacted by IPv4 and IPv6. As developers continue to create new internet applications and services, the transition to IPv6 becomes increasingly crucial for maximizing user reach and leveraging the benefits of IPv6 [5].

Internet Programming: Internet programming, also known as web development, refers to the process of creating applications that run on the World Wide Web. This typically involves writing code in programming languages such as HTML, CSS, JavaScript, and PHP. Internet programming can range from creating simple static websites to developing complex web applications that perform a wide range of tasks. Internet Protocol (IP) Format: The internet protocol (IP) is the principal communications protocol in the Internet protocol suite for relaying data across network boundaries. IP addresses are binary numbers that uniquely identify devices on the Internet. The IP format specifies how these addresses are represented in a standardized way. The most common IP format used today is IPv4, which uses a 32-bit address space and is represented in dot-decimal notation (e.g. 192.168.1.1). However, there is also a newer version called IPv6, which uses a 128-bit address space and is represented in hexadecimal notation (e.g. 2001:0db8:85a3:0000:0000:8a2e:0370:7334).

Table 1. Different fields in an IPv4 header

Field	Size (bits)	Description
Version	4	IP version (4 for IPv4, 6 for IPv6)
Header Length	4	Length of the header in 32-bit words (min 5)
Type of Service	8	Type of service (QoS)
Total Length	16	Total length of the packet (header + data)
Identification	16	Identification of the packet
Flags	3	Fragmentation flags (reserved, don't fragment, etc)
Fragment Offset	13	Offset of the fragment in the original packet
Time to Live	8	Time to live (max number of hops)
Protocol	8	Protocol of the payload (TCP, UDP, ICMP, etc)
Header Checksum	16	Checksum of the header
Source Address	32	Source IP address
Destination Address	32	Destination IP address
Options	variable	Optional fields (e.g. timestamp, security options)
Padding	variable	Padding to align the header on 32-bit boundaries
Field	Size (bits)	Description

Table 2. Key differences between IPv4 and IPv6

Feature	IPv4	IPv6
Address Space	32 bits	128 bits
Address Format	Dotted Decimal	Hexadecimal
Maximum Addressable Nodes	4.3 billion	Approximately 340 undecillion
Address Configuration	DHCP or Static	DHCPv6 or Stateless Autoconfiguration
Header Format	Variable length	Fixed length
Fragmentation	Performed by sender and intermediate routers	Performed only by sender
Security	Optional	Built-in support
Multicast Support	Limited	Better support
Address length	32 bits	128 bits
Address notation	Dotted-decimal notation	Hexadecimal notation
Example address	192.168.1.1	2001:0db8:85a3:0000:0000:8a2e:0370:7334
Number of segments	Four segments	Eight segments
Segment size	8 bits each	16 bits each
Separator character	Period (".")	Colon (":")
Leading zero omission	Omitted	Allowed
Abbreviation	N/A	Double colon ("::")
Loopback address	127.0.0.1	::1

As you can see, IPv4 addresses use a dotted decimal notation format, while IPv6 addresses use a hexadecimal notation format. IPv6 addresses are longer than IPv4 addresses, as they use a 128-bit address space, compared to IPv4's 32-bit address space.

Table 3. Address Comparison

Address size	32 bits	128 bits
Address notation	Dotted decimal, e.g., 192.168.0.1	Colon hexadecimal, e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334
Address space	Approx. 4.3 billion addresses	Approx. 3.4 x 10^38 addresses
Header size	20-60 bytes	40 bytes
Fragmentation	Routers can fragment packets	Hosts must fragment packets
Quality of Service (QoS)	Limited support	Full support
Security	IPsec is optional	IPsec is mandatory

IPv6 provides notable advancements compared to IPv4, especially in terms of address space and security, and is gaining increasing importance as the demand for internet connectivity continues to grow. The format of IPv4 and IPv6 addresses differs due to the discrepancy in their address lengths. IPv4 addresses consist of 32 bits and are expressed in dotted-decimal notation, which entails writing them as four decimal numbers separated by periods. Each decimal number represents 8 bits of the address. For instance, the IPv4 address 192.168.1.1 corresponds to the binary address 11000000.10101000.00000001.00000001. Conversely, IPv6 addresses consist of 128 bits and are expressed in hexadecimal notation. They are written as eight groups of four hexadecimal digits separated by colons (RIPE, 2021). For example, the IPv6 address

2001:0db8:85a3:0000:0000:8a2e:0370:7334 corresponds to the binary address 00100000000000010000110110111000010000101101000000 0000000000000000000000000000000000010001000101101 1110000000000000000000000010001011000000000001110 0 110100.

Due to the surge in Internet usage, IPv4 addresses are becoming scarce, and there are insufficient unique IPv4 addresses to allocate to all devices. IPv6 was introduced to tackle this issue by providing a significantly larger address space, allowing for the allocation of unique addresses to all devices, including mobile devices and Internet of Things (IoT) devices. Overall, the format of IPv4 and IPv6 addresses differs, with IPv4 utilizing dotted-decimal notation and IPv6 employing hexadecimal notation. This disparity has implications for network communication and internet programming, and developers must be mindful of these distinctions when working with IP addresses in their applications.

As we comprehend, IPv4 addresses are expressed in dotted-decimal notation, consisting of four segments of 8 bits each, while IPv6 addresses are expressed in hexadecimal notation, comprising eight segments of 16 bits each. IPv6 addresses also permit leading zero omission and abbreviation through the double colon "::" notation. Additionally, the loopback address differs between IPv4 and IPv6, with IPv4 utilizing 127.0.0.1 and IPv6 employing ::1. All in all, the format of IPv4 and IPv6 addresses differs significantly due to the divergence in their address lengths, which has implications for network communication and internet programming [6].

3. Internet Programming

Internet programming is the creation of applications and software that operate on the internet. It encompasses a broad range of technologies and programming languages, including web development, mobile app development, cloud computing, and Internet of Things (IoT) development. Web development, which involves building websites and web applications, is a common form of internet programming. This can entail front-end development using HTML, CSS, and JavaScript, as well as back-end development using programming languages like PHP, Python, and Ruby. Web developers also collaborate with web servers like Apache and Nginx, as well as databases such as MySQL and MongoDB. Mobile app development entails creating applications that function on mobile devices like smartphones and tablets. This can involve development for Android or iOS using programming languages such as Java, Kotlin, Swift, and Objective-C. Mobile app developers also interact with mobile platforms and APIs such as Google Play and Apple App Store. Cloud computing involves the creation of applications and services that operate on cloud platforms like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud. This may entail the use of cloud-based technologies like serverless computing, containerization, and microservices architecture. IoT development involves the creation of software that operates on Internet of Things devices such as sensors and smart appliances. This can involve the development of embedded software using programming languages like C and C++, as well as working with IoT protocols like MQTT and CoAP. Overall, internet programming is a rapidly evolving domain that necessitates adaptability and proficiency in a wide range of technologies and programming languages.

Common tools and technologies used in internet programming include web development frameworks like Ruby on Rails, Django, and Flask, client-side libraries and frameworks like React, Angular, and Vue.js, and server-side programming languages like PHP, Python, Java, and Node.js. Other crucial concepts in internet programming include database design, web security, and web optimization techniques. Internet programming is a rapidly evolving field, with new technologies and frameworks regularly emerging. As a result, developers in this field must continuously stay updated with the latest trends and best practices to ensure they are utilizing the most efficient tools and techniques to develop their applications.

Internet programming, also known as web programming, is the process of creating and maintaining web applications and websites that can be accessed through the internet. It involves the use of various programming languages, frameworks, and tools to create dynamic and interactive web pages, web services, and web applications that can be accessed by users worldwide. Some common languages used in internet programming include HTML, CSS, JavaScript, PHP, Python, Ruby, Java, and C#. These languages are used to create various components of a website or web application, such as the user interface, server-side logic, database integration, and communication with other web services [7].

3.1 Internet Programming Format

The format of internet programming depends on the specific technologies and tools used for a particular project. However, in general, internet programming involves creating a combination of client-side and server-side code to create dynamic and interactive web pages and applications. Client-side programming involves creating code that is executed by the user's web browser, such as HTML, CSS, and JavaScript. This code is responsible for creating the user interface, handling user input, and executing client-side logic. Client-side programming is often used for creating interactive web pages and user interfaces. Server-side programming involves creating code that is executed on the web server, such as PHP, Python, and Ruby. This code is responsible for processing user requests, executing server-side logic, and interacting with databases and other web services. Server-side programming is often used for creating web applications and web services that provide dynamic and personalized content to users. Internet programming also involves the use of various frameworks, libraries, and tools to streamline the development process and improve the performance of web applications. These tools can include content management systems (CMS) like WordPress and Drupal, e-commerce platforms like Shopify and Magento, and web development frameworks like Ruby on Rails, Django, and Laravel [8].

Table 4. some common components of internet programming

Component	Client-Side Programming	Server-Side Programming
Language	HTML, CSS, JavaScript, TypeScript	PHP, Python, Ruby, Node.js, Java, C#
Execution	Executed by the user's web browser	Executed on the web server
Responsibility	Creating the user interface, handling user input, executing logic	Processing user requests, executing server-side logic, interacting with databases and other web services
Frameworks/ Libraries	React, Angular, Vue.js, jQuery, Bootstrap	Express.js, Flask, Ruby on Rails, Django, Laravel, Spring, .NET

It must be clear that this table is not exhaustive, and there are many other languages, frameworks, and libraries that can be used for internet programming. Additionally, many modern web applications use a combination of client-side and server-side programming to create a seamless user experience.

Table 5. Common components and formats used in internet programming

Component	Format
Hypertext Markup Language (HTML)	Text-based markup language for creating web pages
Cascading Style Sheets (CSS)	Stylesheet language for describing the presentation of HTML and XML documents
JavaScript	High-level programming language for creating interactive web pages
PHP	Server-side scripting language for creating dynamic web pages and web applications
Python	High-level programming language used for web development, data analysis, and artificial intelligence
Ruby	High-level programming language used for web development and scripting
Java	Object-oriented programming language used for creating web applications and mobile apps
C#	Object-oriented programming language used for creating web applications and desktop applications
Content Management Systems (CMS)	Software systems for managing and creating digital content, often used for creating websites and blogs
E-commerce Platforms	Platforms for creating online stores and managing e-commerce operations
Web Development Frameworks	Pre-built libraries, tools, and APIs for streamlining web development and improving application performance

This table highlights some of the most commonly used components and formats in internet programming, but it is by no means an exhaustive list. Internet programming is a diverse and constantly evolving field, and new technologies and tools are constantly emerging to help developers create better and more efficient web applications and websites.

Table 6. some of the key components and technologies used in internet programming

Component/Technology	Description	Examples
Client-side code	Code executed by the user's web browser	HTML, CSS, JavaScript
Server-side code	Code executed on the web server	PHP, Python, Ruby
Web development frameworks	Tools and libraries for building web applications	Ruby on Rails, Django, Laravel
Content management systems (CMS)	Tools for creating and managing website content	WordPress, Drupal, Joomla, Magento
E-commerce platforms	Tools for building and managing online stores	Shopify, Magento, WooCommerce
APIs	Interfaces for interacting with external web services	RESTful APIs, SOAP APIs, GraphQL APIs
Databases	Storage systems for web application data	MySQL, MongoDB, PostgreSQL
Version control	Tools for managing code changes and collaboration	Git, Subversion, Mercurial
Deployment	Process of publishing web applications to production servers	AWS, Heroku, DigitalOcean

Note that this table is not exhaustive, and there are many other technologies and tools used in internet programming depending on the specific needs of a project. The table is meant to provide a general overview of some of the key components and technologies involved in internet programming.
Given table comparing some of the features and characteristics of client-side and server-side internet programming:

Table 7. Key features of client-side and server-side internet programming

Feature	Client-Side Programming	Server-Side Programming
Language	HTML, CSS, JavaScript	PHP, Python, Ruby, Java, C#
Execution environment	User's web browser	Web server
Responsiveness	Quick response time for user input	Slower response time due to server-side processing
User interface	Create dynamic and interactive user interfaces	Limited interaction with user interface
Security	Limited security features	Robust security features
Data storage and retrieval	Limited data storage and retrieval capabilities	Extensive data storage and retrieval capabilities
Application architecture	Single-page applications (SPA), Progressive Web Applications (PWA)	Model-View-Controller (MVC), Representational State Transfer (REST)
Frameworks and libraries	React, Vue, Angular, jQuery	Django, Ruby on Rails, Laravel, Spring Boot
Integration with databases	Limited integration with databases	Extensive integration with databases
Communication with web APIs	Limited communication with web APIs	Extensive communication with web APIs
Application scalability	Limited scalability	Highly scalable
Code execution location	User's web browser	Web server
Primary languages	HTML, CSS, JavaScript	PHP, Python, Ruby, Java, C#, etc.
Functionality	User interface, client-side logic	Server-side logic, database access
Data exchange	Communicates with web services	Communicates with database
Example frameworks	React, Vue, Angular, jQuery	Ruby on Rails, Django, Laravel
Code execution location	Web browser	Web server
Languages used	HTML, CSS, JavaScript	PHP, Python, Ruby, Java, C#
Execution speed	Fast	Slower than client-side
Access to resources	Limited to client	Access to server

	resources	resources
User input handling	Immediate response to user input	Delayed response to user input
User interface	Created using client-side code	Created using a combination of code
Database interaction	Limited to client-side storage	Direct interaction with databases
Frameworks and tools	React, Angular, Vue, jQuery	Express, Django, Ruby on Rails, ASP

As we can see, client-side programming focuses on creating code that is executed by the user's web browser and is responsible for creating the user interface and handling client-side logic. Common languages used in client-side programming include HTML, CSS, and JavaScript, and frameworks such as React, Vue, Angular, and jQuery can be used to help streamline the development process. Server-side programming, on the other hand, focuses on creating code that is executed on the web server and is responsible for processing user requests, executing server-side logic, and interacting with databases and other web services. Common languages used in server-side programming include PHP, Python, Ruby, Java, and C#, and frameworks such as Ruby on Rails, Django, and Laravel can be used to help streamline the development process. Overall, the format of internet programming involves creating a combination of client-side and server-side code to create dynamic and interactive web pages and applications. A variety of languages, frameworks, and tools are used to help streamline the development process and create efficient and user-friendly web applications [9].

3.2 Internet Programming with Internet Protocol Format

Table 8. comparing the key features of internet programming and internet protocol

Feature	Internet Programming	Internet Protocol
Focus	Creating web applications and websites	Managing data transmission over the internet
Languages used	HTML, CSS, JavaScript, PHP, Python, Ruby, Java, C#	IPv4, IPv6
Execution location	Client-side or server-side	Network layer of the OSI model
Data transmission	HTTP, HTTPS, AJAX, WebSocket, WebRTC	TCP, UDP, ICMP, IP
Security	SSL/TLS, encryption, authentication	IPsec, VPN, firewalls, network security tools
Frameworks and tools	CMS (e.g. WordPress), web development frameworks	Wireshark, Ping, Traceroute, Nmap
Error handling	Displaying error messages in web pages	Logging and analyzing network errors
Code execution location	Web browser, Web server	Network devices, End-systems
Data format	HTML, JSON, XML, etc.	Packets, Headers
Transmission mode	Request-response, Publish-subscribe, Streaming, etc.	Connection-oriented, Connectionless
Addressing	URL, IP address	IP address, MAC address
Transport protocols	HTTP, HTTPS, WebSockets, MQTT, etc.	TCP, UDP
Routing	Client-side, Server-side	Routers, Switches
Security	Authentication, Authorization, Encryption, etc.	IPsec, SSL/TLS, Firewalls, Intrusion Detection and Prevention (IDS/IPS)
Purpose	Creating and maintaining web applications and websites	Facilitating communication between devices over the internet
Code execution location	Web browser (client-side), web server (server-side)	Network devices (routers, switches, etc.)
Languages used	HTML, CSS, JavaScript, PHP, Python, Ruby, Java, C#	IPv4, IPv6, TCP, UDP, ICMP
Frameworks and tools	CMS (WordPress, Drupal), e-commerce platforms (Shopify, Magento), web development frameworks (Ruby on Rails, Django, Laravel)	Wireshark, Ping, Traceroute, Nmap, etc.
Protocols used	HTTP, HTTPS, FTP, SMTP, POP3, IMAP, etc.	IPv4, IPv6, TCP, UDP, ICMP
Transmission	Information is transmitted through web servers and client devices using a variety of protocols	Information is transmitted through network devices using Internet Protocol
Code execution location	Web browser and web server	Network devices and computers
Languages used	HTML, CSS, JavaScript, PHP, Python, Ruby, Java, C#	IP, TCP, UDP, HTTP, FTP, SMTP, DNS
Execution speed	Depends on the implementation	Generally fast and efficient
Access to resources	Limited to web resources	Access to network resources
User input handling	Immediate response to user input	Delayed response to network traffic
User interface	Created using client-side code and server-side code	Not applicable
Database interaction	Direct interaction with databases	Not applicable
Frameworks and tools	React, Angular, Vue, jQuery, Express, Django, Ruby on Rails, ASP	Wireshark, Ping, Traceroute, Nmap
Purpose	Develop web applications, websites	Provide a standardized method for data transmission over the internet
Languages used	HTML, CSS, JavaScript, PHP, Python, Ruby, Java, C#	IP, TCP, UDP, ICMP
Execution location	Client-side and server-side	Network layer
Data transfer method	HTTP, FTP, SMTP, WebSocket, etc.	IP, TCP, UDP
Data format	HTML, XML, JSON, etc.	Binary, IPv4, IPv6, ICMP, etc.
Reliability	Depends on the application	TCP provides reliable data transfer
Speed	Depends on the application	High-speed data transfer over the internet
Security	HTTPS, SSL/TLS, etc.	IPsec, VPN, etc.

As we know, internet programming is focused on creating web applications and websites using various programming languages and tools. It involves creating client-side and server-side code to create dynamic and interactive web pages and web applications. On the other hand, internet protocol is focused on managing data transmission over the internet. Internet protocol uses the IP addressing scheme, either IPv4 or IPv6, to route data packets over the internet using protocols such as TCP, UDP, and ICMP. It also involves implementing security measures such as IPsec, VPN, firewalls, and other network security tools to protect data transmission. Internet programming can use various data transmission protocols such as HTTP, HTTPS, AJAX, WebSocket, and WebRTC to communicate with web servers and other web services. It also involves implementing security measures such as SSL/TLS encryption and authentication to protect user data. Both internet programming and internet protocol involve the use of frameworks and tools to streamline the development process and improve performance. For example, internet programming uses content management systems (CMS) like WordPress and web development frameworks like Ruby on Rails and Django, while internet protocol uses network analysis tools like Wireshark, Ping, Traceroute, and Nmap [10].

Table 9. comparing the features of client-side and server-side internet programming with the internet protocol version IPv4 and IPv6

Feature	Client-side Programming	Server-side Programming	IPv4	IPv6
Code execution location	Web browser	Web server	N/A	N/A
Languages used	HTML, CSS, JavaScript	PHP, Python, Ruby, Java, C#	N/A	N/A
Execution speed	Fast	Slower than client-side	Dependent on network bandwidth	Dependent on network bandwidth
Access to resources	Limited to client resources	Access to server resources	Access to local network	Access to global network
User input handling	Immediate response to user input	Delayed response to user input	N/A	N/A
User interface	Created using client-side code	Created using a combination of code	N/A	N/A
Database interaction	Limited to client-side storage	Direct interaction with databases	N/A	N/A
Frameworks and tools	React, Angular, Vue, jQuery	Express, Django, Ruby on Rails, ASP	N/A	N/A

Client-side programming and server-side programming are not directly related to the internet protocol version, as they are programming concepts that relate to web development [11]. However, both client-side and server-side programming interact with the internet protocol in order to communicate between the client and server. The internet protocol version affects the way data is transmitted across the internet, including the speed of execution and access to resources. IPv4 and IPv6 also have different address formats, with IPv4 using 32-bit addresses and IPv6 using 128-bit addresses. Internet programming can interact with IPv4 and IPv6 addresses in order to communicate with other devices and services across the internet. For example, web applications can use APIs or other web services to communicate with databases or other web applications on remote servers. Frameworks and tools for internet programming can also provide support for working with IPv4 and IPv6 addresses, including libraries and APIs for handling network communication and socket programming [12].

4. Pros and Cons of using Internet Protocol (IP) with Internet Programming (IP)

Table 10. Pros and Cons

Pros.	Cons.
IP is a widely-used protocol that has been around for decades, which means it is well-established and supported by a wide range of software and hardware.	IP can be vulnerable to security threats, such as IP spoofing and packet sniffing.
IP is highly reliable and can handle large volumes of data traffic.	IP relies on the correct configuration of network devices and routers, which can be complex and difficult to manage in large-scale networks.
IP is a flexible protocol that can be used for a variety of applications, including web browsing, email, file sharing, and more.	IP addresses can be difficult to remember and manage, particularly as the number of connected devices continues to grow.
IP is compatible with a wide range of other internet protocols, including TCP, UDP, and ICMP.	The proliferation of IPv4 addresses has led to a shortage of available addresses, which can limit the growth and scalability of internet-based applications.
IP is a widely-used protocol that has been around for decades, which means it is well-established and supported by a wide range of software and hardware.	IP can be vulnerable to security threats, such as IP spoofing and packet sniffing.
Compatibility: IP is the backbone of the internet and is used to route data packets across the network. As such, it is a widely accepted and compatible protocol that can be used across different operating systems, devices, and applications.	Complexity: The IP protocol is a complex protocol that requires a deep understanding of networking concepts and technologies. This can make it difficult for beginners to work with and troubleshoot issues.
Flexibility: IP is a flexible protocol that can be used for a wide range of applications, including file transfer, email, voice and video calls, and web browsing, among others.	Reliability: The reliability of IP can be impacted by various factors, such as network congestion, packet loss, and routing errors, which can lead to slower transmission speeds or data loss.
Security: IP includes several security features, such as authentication and encryption, that can help protect data transmitted over the internet from unauthorized access and interception.	Security: Although IP includes security features, it is still vulnerable to attacks such as IP spoofing, which can lead to data breaches and other security issues.
Widely used: The Internet Protocol is a widely used protocol for transmitting data over the internet. This means that if you are building internet applications or software, you will likely be working with IP, and having a good understanding of it can help you build more efficient and effective applications.	Limited security: While the Internet Protocol is reliable, it does not provide robust security measures on its own. Additional security measures, such as encryption and firewalls, must be implemented to ensure that data transmitted over IP is secure.
Standardized: The Internet Protocol is standardized, meaning that it is well-defined and consistent across different implementations. This makes it easier to work with and ensures that your applications will be compatible with other systems that use IP.	Complexity: The Internet Protocol can be complex to work with, especially for programmers who are new to internet programming. This complexity can make it challenging to troubleshoot issues that arise when working with IP.
Scalable: The Internet Protocol is designed to be scalable, meaning that it can handle large volumes of data and is able to accommodate changes to the network as it grows.	Network dependence: The Internet Protocol is dependent on the network infrastructure of the internet, which can lead to performance issues if the network is congested or not functioning optimally.
Reliable: The Internet Protocol is designed to be reliable, meaning that it ensures that data is	Fragmentation: The Internet Protocol can fragment data packets if they are too large to be transmitted in a single

transmitted securely and arrives at its destination without corruption.	packet. Fragmentation can cause delays and inefficiencies in data transmission and can make it challenging to reassemble the data at the receiving end.
Universal compatibility: IP is a universal protocol that is used by all devices connected to the internet. This makes it an essential component of internet programming, as it ensures that your applications and software will be able to communicate with other devices on the internet.	Security: IP is a relatively insecure protocol, as it does not provide built-in encryption or security features. This can make it vulnerable to hacking, data breaches, and other security threats. As a result, internet programmers need to take extra steps to ensure the security of their applications and software when working with IP.
Flexibility: IP is a flexible protocol that can be used for a wide range of applications, from simple web browsing to complex data transfer and communication. This flexibility makes it an ideal protocol for internet programming, as it allows developers to create a wide range of applications and software that can be used across multiple platforms.	Reliability: IP is a best-effort protocol, which means that it does not guarantee the delivery of data packets. This can result in lost or corrupted data packets, which can impact the reliability of internet applications and software.
Universal standard: IP is a universal standard for transmitting data across the internet, and it is used by all devices connected to the internet. This makes it an essential protocol for internet programming. Scalability: IP is highly scalable and can handle large amounts of data traffic, making it ideal for developing internet applications that require high levels of data transmission. Flexibility: IP can be used with a variety of other internet protocols, including TCP, UDP, and ICMP, among others, giving programmers a lot of flexibility when developing internet applications.	Security: IP is not inherently secure, and it does not provide any encryption or authentication mechanisms. This means that additional security measures must be implemented to ensure that data transmitted over IP is secure. Complexity: IP is a complex protocol, and it requires a high level of technical expertise to work with effectively. This can make it challenging for developers who are new to internet programming. Fragmentation: IP packets can be fragmented when they are transmitted across the internet, which can cause delays and errors in data transmission. This can be particularly problematic for internet applications that require real-time data transmission.

While IP is a reliable and widely-used protocol in Internet programming, it does have some potential drawbacks, particularly in terms of security and scalability. However, by understanding these challenges and implementing appropriate security measures and network management strategies, programmers can effectively utilize IP and other internet protocols to develop robust and efficient internet-based applications. While Internet Protocol is a crucial component of internet programming, it does come with certain challenges, such as security concerns and complexity. However, the benefits of using IP, such as its universal standard, scalability, and flexibility, make it an essential protocol for internet programming [13], [14].

5. Internet Protocol with Internet Programming Architecture and Design

When designing the architecture of an internet programming mission that uses Internet Protocol (IP), it is important to do not forget several key components:

- Network layer: IP operates on the community layer of the OSI model and is accountable for routing statistics packets throughout the internet. The structure of the utility ought to don't forget the community topology and the interactions among gadgets at the community, ensuring that IP is used efficiently to transmit records.
- Transport layer: The shipping layer is accountable for coping with the communication between the devices using IP. This layer consists of protocols along with TCP or UDP that provide dependable or unreliable facts transfer respectively. In addition to IP, the shipping layer protocols such as TCP, UDP or SCTP, amongst others, are used in internet programming to offer additional capability which includes reliable transmission, glide manage, and congestion avoidance. The architecture have to don't forget the interactions between these protocols and make sure that they may be used efficiently to satisfy the desires of the utility.
- Application Layer: The top layer of the protocol stack this is answerable for presenting facts to the person or application. In internet programming, this accretion usually includes protocols along with HTTP, FTP, and SMTP. The software layer of the architecture must encompass the software program programs that are designed to apply IP. These packages will commonly use different protocols, which include HTTP or FTP, in conjunction with IP to permit communication between the distinctive devices.
- Security layer: Security is a key issue in internet programming, and the architecture ought to consist of security mechanisms to make certain the confidentiality, integrity, and availability of facts. Security mechanisms inclusive of encryption, authentication, and access controls need to be applied to protect the facts this is transmitted over the network. As mentioned before, IP isn't always inherently stable and extra security measures may need to be applied to make certain that information transmitted over IP is covered. The architecture ought to encompass measures together with encryption, authentication, and get admission to controls to make sure that records is transmitted securely.
- Server and consumer structure: The architecture of the application must recollect the relationship among servers and clients, which includes the usage of internet servers, software servers, and databases. The architecture need to additionally recall factors together with load balancing and caching to optimize overall performance and scalability.
- Application programming interface (API): An API is regularly used in internet programming to provide a standardized manner for applications to speak with every different. The architecture ought to bear in mind the design and implementation of the API, making sure that it meets the desires of customers and is well matched with other programs.
- API structure: Internet programming tasks often use an API (Application Programming Interface) structure, in which distinctive components of the utility communicate with each other thru APIs. IP can be used to course data among different components of the utility that use APIs.

- Network infrastructure: The community infrastructure is a key component of the architecture as it presents the physical and logical infrastructure required for the conversation between the devices. The network infrastructure should consist of network gadgets along with routers, switches, and firewalls that facilitate the communication between the gadgets using IP.
- Internet layer: The Internet layer is the layer where IP is used. This layer provides routing of data packets throughout the internet, ensuring that they attain their supposed vacation spot.
- Network topology: The architecture need to bear in mind the network topology and make sure that IP is used successfully to transmit records throughout the network. The structure should be designed to handle unique types of community configurations, which include LANs, WANs, and the net.
- Protocol stack: The architecture have to include a protocol stack that makes use of IP and different protocols efficaciously. This may also include protocols such as TCP, UDP, ICMP, HTTP, and others, relying at the wishes of the software. The architecture of the challenge must additionally do not forget the various protocols in an effort to be used inside the protocol stack, which includes TCP, UDP, and HTTP. The structure have to make certain that those protocols are used effectively and in a way that meets the desires of the application.
- Scalability: The structure of the assignment need to be designed to deal with multiplied visitors and statistics volumes, at the same time as maintaining overall performance and protection. This can also involve designing the architecture with concerns which include clustering, sharding, and horizontal scaling. This may also encompass strategies such as load balancing, clustering, and cloud computing.
- Performance: The architecture ought to be optimized for performance, taking into account factors consisting of network latency, bandwidth, and processing pace. This may additionally encompass strategies such as caching, compression, and content material shipping networks.
- Error coping with: The architecture have to consist of blunders coping with mechanisms which can detect and get over mistakes within the transmission technique. This may include strategies including errors correction codes and retransmission protocols.
- Client-server architecture: Most net programming initiatives use a purchaser-server architecture, in which the purchaser makes requests to a server, and the server responds with the asked facts. IP plays a critical role on this structure with the aid of routing facts packets among customers and servers.
- Distributed architecture: In a few instances, net programming initiatives may also use a disbursed structure, where multiple servers paintings collectively to procedure requests and offer data to clients. IP can be used to course facts between one of a kind servers in a dispensed structure. Distributed architecture may be essential to make certain that statistics can be transmitted effectively and reliably across the network. This can involve the use of load balancing, content transport networks (CDNs), and other techniques to distribute statistics and processing across more than one servers.
- Layered architecture: Internet programming projects frequently use a layered structure, wherein one-of-a-kind layers of the application are chargeable for distinctive obligations. For instance, the software may additionally have a presentation layer, a business good judgment layer, and a information access layer. IP can be used to direction information between one-of-a-kind layers of the software.
- Microservices architecture: Internet programming projects may use a microservices architecture, wherein special parts of the utility are damaged down into smaller, independent offerings. IP may be used to path records between distinct microservices.
- Redundancy: Finally, the structure of the mission ought to consist of redundancy measures to make certain that the application remains available in the event of a failure. This may involve designing the architecture with considerations such as failover, replication, and disaster recovery.

The architecture of an internet programming project that uses IP should take into account factors such as the network and transport layers, security, server and client architecture, and API design to ensure that the application meets the needs of users and functions effectively on the internet. A well-designed internet programming architecture that includes IP should take into account the requirements of the different layers, including the network infrastructure, application layer, transport layer, internet layer, physical layer, and security layer. This will ensure that the architecture is scalable, reliable, and secure, and can support the communication requirements of the different devices and applications that are connected to the internet [15], [16].

For example, the architecture of a web application that uses IP might include a front-end layer that communicates with the user's web browser using HTTP, a middle layer that communicates with a database server using TCP, and a back-end layer that communicates with other servers on the internet using IP. In addition, the architecture should take into account issues such as security, with measures such as encryption and authentication to protect data transmitted over IP, scalability, with measures such as load balancing and clustering to handle increased traffic, and performance, with measures such as caching and compression to optimize data transmission. The architecture for an internet programming project that uses IP should be carefully designed to ensure that it meets the needs of the application and provides a secure and efficient platform for data transmission. This may involve considerations such as network topology, the protocol stack, distributed architecture, security, and performance optimization [17], [18].

6. Conclusion

Internet programming includes growing packages and software that operate on the net or are on hand thru internet browsers. This form of programming requires knowledge of internet protocols, which include the internet protocol (IP) format. The IP format is a standardized set of rules that govern the transmission of records across the internet. It includes a header segment and a facts section, which contain statistics approximately the sender, receiver, and content of the facts being transmitted. Understanding the IP layout is critical for developing programs and software program which can communicate over the net. As a programmer, it's far important to be familiar with the extraordinary fields in an IP header and their corresponding descriptions. This information will allow us to develop extra efficient and powerful programs that may transmit facts over the net securely and reliably. Overall, the aggregate of internet programming and knowledge of the IP layout is vital for growing cutting-edge, net-based totally programs that may be used by human beings all over the world.

In conclusion, the Internet Protocol (IP) is a fundamental protocol in net verbal exchange that permits the transmission of statistics between gadgets over the net. As a programmer operating with the net, it's far crucial to understand the format and shape of IP packets to correctly increase packages and software program that could transmit and obtain facts over the internet. As a programmer, it's miles critical to understand the IP layout and how it's miles used in net programming. This understanding can help us troubleshoot issues and optimize performance when growing programs for the internet. The development of internet packages and software program requires adherence to a variety of internet protocols, no longer just IP. These encompass protocols along with HTTP, TCP, and DNS, amongst others. A appropriate expertise of those protocols is essential to constructing robust, efficient, and stable internet programs. Internet programming requires a solid understanding of internet protocols, such as the Internet Protocol (IP). The IP protocol plays a essential function in transmitting statistics throughout the net, and information its format and the way it's far used can assist ensure that your internet applications and software program function nicely and securely.

Data Availability

The data supporting findings of this study are all presented in the article.

Conflict of Interest

The authors declare no conflicts of interest.

Funding Source

There is no funding source for this paper.

Authors' Contributions

Author-1, Arun Kumar Singh, researched the literature, collected data and organized the study. Author 2 supervised the preparation of the study and gave final approval of the article for publication.

Acknowledgements

We appreciate and thank the International Journal of Scientific in Research in Computer Science and Engineering (isroset) for their assessment and suggestions to improve visibility and clarity.

References

[1] Sobin, "A survey on architecture, protocols and challenges in IoT", Wireless Personal Communication, 1(12), pp 1383–1429, 2020

[2] Seyed Mostafa Bozorgi, Mehdi Golaorkhtabaramiri Samaneh Yazdanid, "A Smart Optimizer Approach for Clustering Protocol in UAV-assisted IoT Wireless Networks", Journal of Internet of Things; 2023

[3] J. Chen, C. Touati, Q. Zhu, "Optimal Secure Two-layer IoT Network Design", IEEE Transaction Control Networking Systems, 7(1), pp 398–409, 2020.

[4] Andrew L. Russell, "Open standards and the digital age: history, ideology, and networks", New York: Cambridge Univ Press. pp. 196, 2014.

[5] Janet Abbate, "Inventing the Internet", MIT Press, ISBN 978-0-262-51115-5, pp. 123–4, 2000.

[6] Lin, Yi-Bing, Min-Zheng Shieh, "To Learn Programming through Internet of Things." IEEE Internet of Things Magazine 5, no. 4 (December 2022): pp.168–72, 2022

[7] N. M. Garcia, P. P. Monteiro, M. M. Freire, "Burst assembly with real IPv4 data–performance assessment of three assembly algorithms," in International Conference on Next Generation Wired/Wireless Networking, 2006, pp. 223–234, 2006.

[8] Q. Wang et al., "Multimedia IoT systems and applications," in 2017 Global Internetof Things Summit (GIoTS), 2017, pp. 1–6, 2017

[9] S. Bj, "Can OTN be replaced by Ethernet? A network level comparison of OTN and Ethernet with a 5G perspective," in 2018 International Conference on Optical Network Design and Modeling (ONDM), pp. 220–225, 2018

[10] S. Zaman S., F. Karray. Fuzzy ESVDF approach for Intrusion Detection System. The IEEE 23rd International Conference on Advanced Information Networking and Applications (AINA-09). May 26-29, 2009.

[11] I. Onut, A. Ghorbani. "A Feature Classification Scheme for Network Intrusion Detection", International Journal of Network Security, pp.1-15,2007.

[12] A. Tamilarasan, S. Mukkamala, A. Sung, K. Yendrapalli, "Feature Ranking and Selection for Intrusion Detection Using Artificial Neural Networks and Statistical Methods", 2006 International Joint Conference on Neural Networks (IJCNN'06), pp.4754-4761, 2006.

[13] A. Sung, S. Mukkamala, "Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks", Symposium on Application and Internet (SAINT'03), pp.209-216, 2003.

[14] Amit Shukla, H.K. Application of robotics in offshore oil and gas industry—A review part II. Robot. Auton. Syst. 2015, 75, pp.508–524, 2015.

[15] E. Brucherseifer, H. Winter, A. Mentges, M. Muehlhauser, M. Hellmann, "Digital Twin conceptual framework for improving critical infrastructure resilience", pp.1062–1080, 2021.

[16] P. Laplante, B. Amaba, "Artificial Intelligence in Critical Infrastructure Systems", Computer 2021, 54, pp.14–24, 2021

[17] Y. Wu, H.N. Dai, H. Wang, "Convergence of Blockchain and Edge Computing for Secure and Scalable IIoT Critical Infrastructures in Industry 4.0", IEEE Internet Things J. 2020, 8, 2300–2317, 2020.

[18] S. Park, S. Kwon, Y. Park, D. Kim, I. You, "Session Management for Security Systems in 5G Standalone Network", IEEE Access 2022, 10, pp.73421–73436, 2022.

AUTHORS PROFILE

Arun Kumar Singh currently working as Head of Computer Science in School of Mathematics and Computer Science, PNG University of Technology, Lae, Papua New Guinea. He received B. E. degree in ECE from Agra University, Agra, M. Tech. Degree in IT-WCC from IIIT-Allahabad, Prayagraj, India and PhD degree from SU Meerut, India. He has published more than 50 research papers in reputed international journals and conferences including IEEE and it's also available online. His main research work focuses on Cybersecurity, AI, Sustainability, IoT and Computational Intelligence. He has 20 years of teaching experience and 15 years of research experience.

Alak Kumar Patra earned his B.E. in Civil Engineering from Jadavpur University, Kolkata in 1999, M.E. in Structural Engineering from IIEST Shibpur in 2012, and Ph.D. in Structural Engineering from IIT Kharagpur in 2018. He is currently working as PG CorseCoordinator and Chairman of Research Development and Engagement Committee, in the School of Civil Engineering (SCE), The Papua New Guinea University of Technology since 2022. He is a life time fellow member (F.I.E) of The Institute of Engineer (India). He has more than 28 years' experience in teaching, research and industries.