ProtoPlay: A Tool for Visualizing Network Protocols and Behaviors

Submitted By

Sonia yadav & Tavaheed Tariq

2022BITE030 & 2022BITE008

soniayadavnitsgr@gmail.com & tavaheed_2022bite008@nitsri.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

NATIONAL INSTITUTE OF TECHNOLOGY SRINAGAR

Contents

List of Figures

Chapter 1

Introduction

Network simulation is a technique whereby a software program replicates the behavior of a real network. This is achieved by calculating the interactions between the different network entities such as routers, switches, nodes, access points, links, etc [16]. A network simulator is a software program that can predict the performance of a computer network or a wireless communication network. Since communication networks have become tool complex for traditional analytical methods to provide an accurate understanding of system behavior, network simulators are used. In simulators, the computer network is modeled with devices, links, applications, etc., and the network performance is reported. [16]. To simulate the working or network simulator, we proposed a model `ProtoPlay`, a real-time network simulation tool for visualizing the packet transfer and protocols used in the TCP/IP model [4]. Our source code is publicly available at `https://github.com/Tawheed-tariq/ProtoPlay`.

1.1 Project Scope

ProtoPlay is structured into five layers just like TCP/IP mdoel [4]. The 5 layers consist of Physical Layer, DataLink Layer, Network Layer, Transport Layer and Application Layer. Each layer's functionality is shown in our model and protocols used in these layers are also simulation. ProtoPlay includes simulations for protocols for every layer including `CRC` [11], `CSMA/CD` [2], `Go-Back N` [14], `Stop And Wait` [13] for DataLink Layer and protocols like `IP`, `ARP` [17] and others in Network Layer including other Transport and Application Layer protocols providing user a great experience in visualizing the protocols used in TCP/IP model [4].

Chapter 2

Features

The Network Simulator includes several features that make it a valuable tool for learning network concepts:

2.1 Layer Simulation

The Layer Simulation element simulates all the devices from physical to application layer including bridges, switches, hubs, routers, and End devices. It delivers visualization of the topology of the network utilizing PyVis [9] and NetworkX [6] to ensure users gain good knowledge about the working of the devices. It incorporates real-time visualization of traffic flow through networks to ensure better comprehension by the users on the flow of data within a network. Besides, it also provides analysis of network performance parameters to provide users with information regarding the effectiveness of various network arrangements.

2.2 Protocol Simulation

The Protocol Simulation element applies different network protocols with feedback displays. The implementation of the Cyclic Redundancy Check (CRC) [11] has noise channels that illustrate error detection features. Visualizing Carrier Sense Multiple Access with Collision Detection (CSMA/CD) [2] indicates how devices contend for a shared medium and how they recover from collisions within a network. Stop-and-Wait Protocol illustrates an easy flow control technique with a feedback display explaining how acknowledgments are performed. The Go-Back-N Protocol has an implementation that provides a sliding window protocol with packet transmission visualization and acknowledgment procedures. Protocols like IP, ARP, HTTP, TCP , UDP, FTP and other transport and application layer protocols are also used and provided a good experience to user for simulation of these protocols.

2.3 User Interface

The Network Simulator has an interactive web-based interface developed with Streamlit. [12] The interface gives users real-time visualizations of network operations to allow them to grasp how networks operate. Also we use some python libraries inlcuding [6, 9], Our base line model was based on [14] from which we used the idea for creating user-friendly UI and protocol simulation strategies, for debugging purposes [1] was used as a debugging tool to tackle errors or mistakes make in building the model. Users can configure simulation parameters easily to test various network scenarios. The interface displays simulation results and metrics in a simple

way, and it is easy for users to analyze network performance. The design emphasizes giving a seamless and intuitive experience for learning.

Chapter 3

System Architecture

The Network Simulator follows a modular architecture that separates different components of the system for better maintainability and extensibility.

3.1 Overall Architecture

The system architecture comprises a number of key components collaborating to offer an end-to-end simulation environment. The User Interface Module, which is developed based on Streamlit [12], manages user input and presents simulation outputs in a user-friendly manner. The Network Topology Module is used to generate and control network topologies based on NetworkX [6] for flexible network designs. The Visualization Module is responsible for the visualization of network topologies and protocol activities with PyVis [9], which offers intuitive visual feedback to users. The Protocol Simulation Module simulates various network protocols and their behaviors, enabling users to see how these protocols work. The Layer Simulation Module simulates the behavior of 5 layered TCP/IP protocol stack [4,10], which informs users how these building blocks work within a network.

3.2 Layer wise analysis

ProtoPlay provides simulation for all 5 layers of TCP/IP protocol stack. Each layer implementation is explained below:

3.2.1 Physical Layer

At the physical layer, we worked on creating end devices and hubs, establishing connections between them to form a network topology. This setup enabled the transmission and reception of data across the network, simulating the basic functionality of the physical infrastructure in a communication system. In Fig 3.1 we have shown a basic hub-switch topology in which we have used 2 switches connected to hub and 3 end devices connected to both switches.

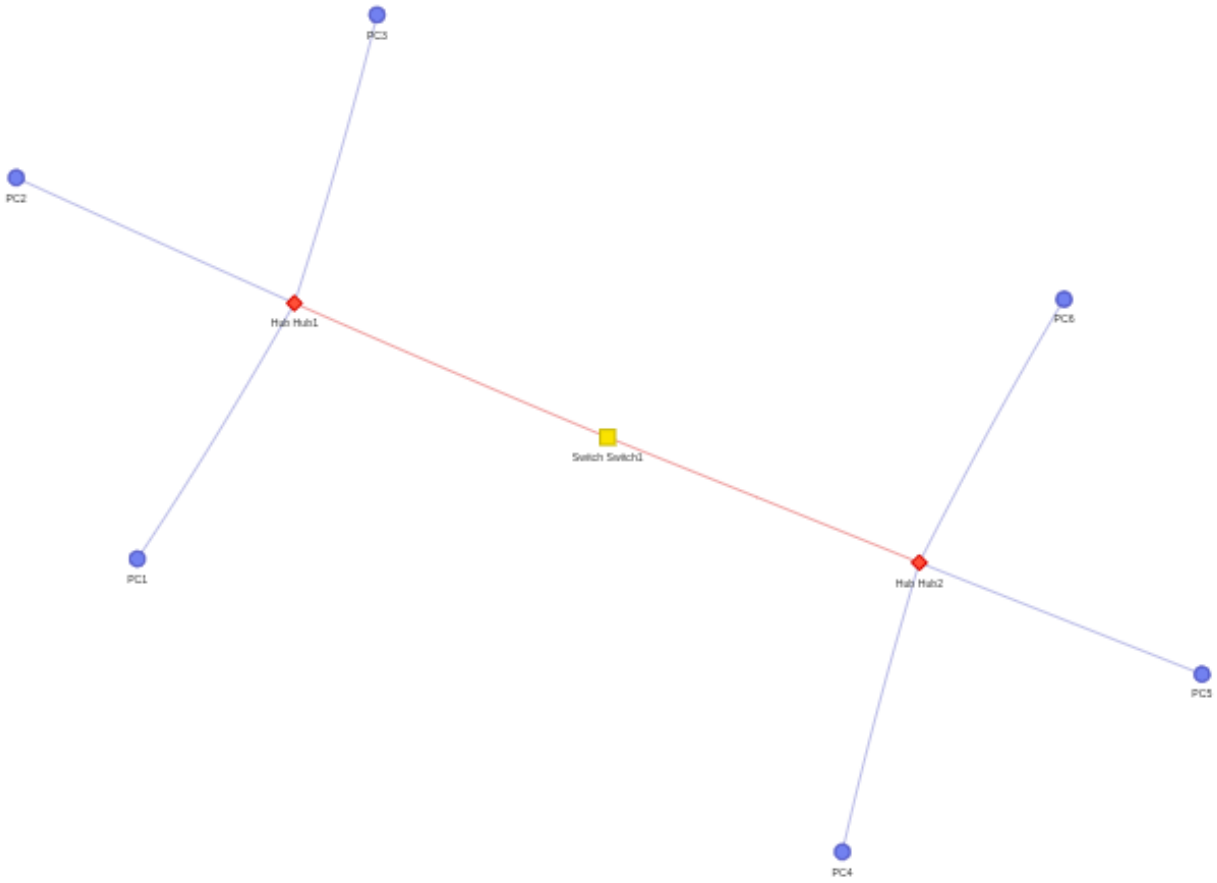

Figure 3.1: A basic Hub-Switch topology with 3 end devices connected to each switch and both switches connected to hub

3.2.2 DataLink Layer

At the data link layer, we focused on creating Layer 2 devices such as bridges and switches. We implemented key functionalities including address learning in switches, CRC-based error detection with noise models [11], and CSMA/CD for medium access control [2]. Additionally, we developed and tested flow control protocols including Stop-and-Wait and Go-Back-N [14]. As part of validation, we designed and tested multiple scenarios. In one test case, we created a switch connected to five end devices and successfully demonstrated data transmission along with all the implemented flow control protocols. We also analyzed and reported the number of broadcast and collision domains in the network. In another case, we constructed two star topologies, each with five end devices connected to a hub, and linked the hubs using a switch to enable communication among all ten devices. Again, we evaluated the number of broadcast and collision domains. The tests confirmed the correctness and effectiveness of our protocols and network configurations.

3.2.3 Network Layer

At network Layer, we successfully implemented key Network Layer functionalities in our simulator. We created and configured routers, assigned well-structured classfull IPv4 addresses to all devices, and used the Address Resolution Protocol (ARP) [17] to map IP addresses to MAC addresses within the local network. Static routing was configured manually to enable

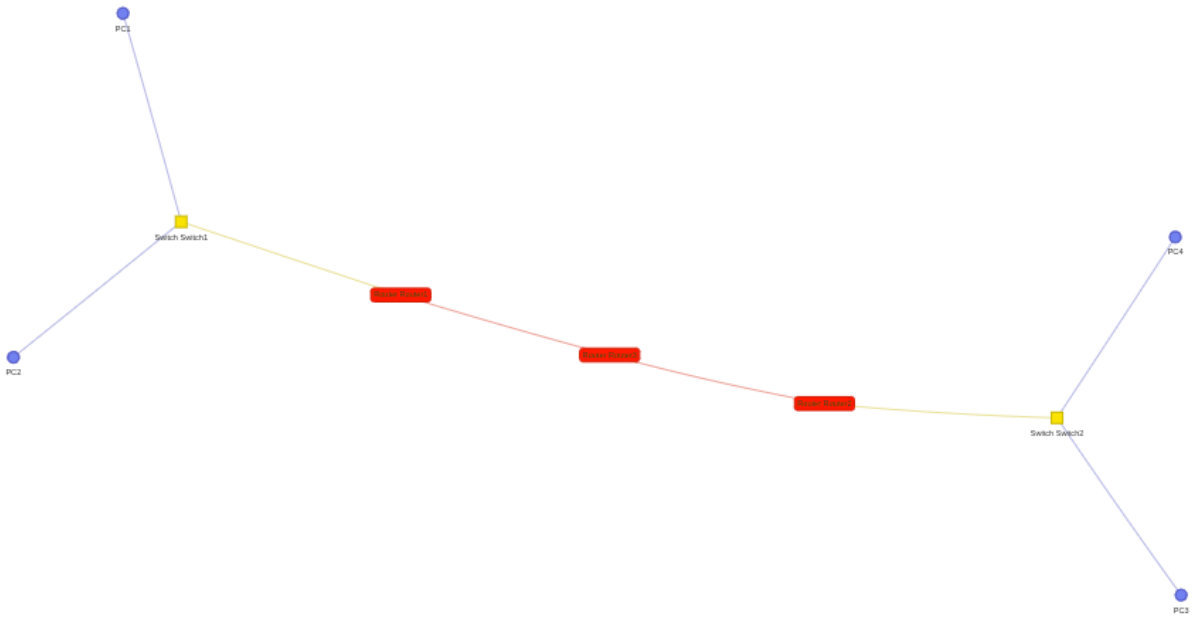

Figure 3.2: Hop Router topology representing 3 routers connected to each other with 2 switches connected to 2 routers at end and both switches contain 2 end devices for packet transfer

inter-network communication based on the longest prefix match [5], while dynamic routing was achieved by implementing the Routing Information Protocol (RIP) [8], allowing routers to exchange routing information and update paths automatically. We verified the correctness of these implementations through various test cases, including ARP resolution, packet forwarding between different networks, and dynamic updates in routing tables when network changes occur. In Fig 3.2 we show the topology for Hop router with 3 routers connected to each other and 2 switches connected at the end of the routers with each switch connected to 2 end devices, showcasing the capability of our model to transfer packets from router to router using static routing as well as dynamic routing.

3.2.4 Transport Layer

For Transport Layer, we implemented essential Transport Layer functionalities in our simulator. We assigned port numbers to various processes, including both well-known and ephemeral ports [15], to enable reliable process-to-process communication. To manage data flow efficiently and handle transmission errors, we integrated a sliding window flow control protocol specifically, the Go-Back-N protocol reused and adapted from our Data Link Layer implementation. This ensures proper sequencing and retransmission of lost packets at the Transport Layer level, demonstrating core TCP-like behavior and supporting smooth end-to-end data delivery. We introduced different transport layer protocols like TCP and UDP [3, 7] passing different test cases performed on the model, hence showcasing a great tool for port assignment and handling complex protocols easily.

3.2.5 Application Layer

In Application Layer, we implemented key Application Layer services that demonstrates real-world network applications. We set up HTTP , FTP and DNS protocols to efficiently transfer data from client and server through these protocols. These services validate the complete

protocol stack, showing how user-level applications interact with lower layers for reliable communication.

Chapter 4

Conclusion

The Network Simulator in this report offers a robust tool for visualization and comprehension of network protocols and behaviors. By providing interactive simulation for protocols and 5 layered TCP/IP protocol stack, the simulator is a valuable educational tool for students and professionals. Through the use of technologies like NetworkX [6], PyVis [9], and Streamlit [12], the simulator delivers a smooth and interactive user experience, while maintainability and extensibility are guaranteed through its modular design.

Bibliography

[1] Claude AI. Claude ai. https://claude.ai/, 2025. Accessed: 2025-03-24.

[2] Yahya Y. Al-Salqan, Cheoul-Shin Kang, James H. Herzog, and E. K. Park. A simulation study of a csma/cd with connected data links. *SIGSIM Simul. Dig.*, 21(3):292–300, April 1991.

[3] Amitha353. The bits and bytes of computer networking. https://github.com/Amitha353/The-Bits-and-Bytes-of-Computer-Networking, 2025. Accessed: 2025-06-15.

[4] Behrouz A. Forouzan and Sophia Chung Fegan. *TCP/IP Protocol Suite*. McGraw-Hill Higher Education, 2nd edition, 2002.

[5] NetFPGA Project. Netfpga public repository. https://github.com/NetFPGA/NetFPGA-public, 2025. Accessed: 2025-06-15.

[6] NetworkX. Networkx documentation. https://networkx.org/, 2025.

[7] pandax381. microps: A minimal tcp/ip protocol stack implementation in python. https://github.com/pandax381/microps, 2025. Accessed: 2025-06-15.

[8] Puetsua. pyrip: Python implementation of routing information protocol. https://github.com/puetsua/pyRIP, 2025. Accessed: 2025-06-15.

[9] PyVis. Pyvis documentation. https://pyvis.readthedocs.io/en/latest/, 2025.

[10] sachinites. Tcp/ip stack implementation. https://github.com/sachinites/tcpip_stack, 2025. Accessed: 2025-06-15.

[11] John S. Sobolewski. *Cyclic redundancy check*, page 476–479. John Wiley and Sons Ltd., GBR, 2003.

[12] Streamlit. Streamlit documentation. https://docs.streamlit.io/, 2025.

[13] Sushma07. Stop and wait protocol implementation. https://github.com/Sushma07/stop-and-wait, 2025. Accessed: 2025-06-15.

[14] Mir Tasleem. Network simulator. https://github.com/Mir-Tasleem/Network-Simulator, 2025.

[15] VasanthVanan. Computer networking: Transport layer notes. https://github.com/VasanthVanan/computer-networking-top-down-approach-notes/blob/main/Chapter%203:%20Transport%20Layer.md, 2025. Accessed: 2025-06-15.

[16] Klaus Wehrle, Mesut Günes, and James Gross. *Modeling and Tools for Network Simulation*. Springer Science & Business Media, 2010. Accessed: 2025-06-15.

[17] Wikipedia contributors. Address resolution protocol — wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Address_Resolution_Protocol, 2025. Accessed: 2025-06-15.