Network Design Automation in Cisco Packet Tracer*

1st Hasan Abbas
Reg No: 2022204)
u2022204@giki.edu.pk

2nd Muhammad Murtaza
Reg No: 2022402)
u2022402@giki.edu.pk

3rd Shaheer
Reg No: 2022424
u2022424@giki.edu.pk

***Abstract*—This paper presents an innovative approach to automating network design in Cisco Packet Tracer through artificial intelligence and natural language processing. We develop a pipeline that translates natural language descriptions into functional network configurations, significantly reducing the technical barriers in network simulation. By reverse engineering the Packet Tracer file format and implementing a dual-LLM architecture, our system successfully generates valid network topologies from user descriptions. The solution demonstrates high accuracy in component identification, connection mapping, and IP address allocation. While currently limited to Packet Tracer version 5, our approach establishes a foundation for more accessible network design tools. Results show successful automation of basic to moderate network configurations, with potential applications in educational settings and professional training environments.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

The increasing complexity of computer networks has made simulation tools essential for education and design validation. Cisco Packet Tracer stands as a leading network simulation platform, yet its utilization requires significant technical expertise, creating barriers for beginners and non-specialists. This research addresses this challenge by developing an AI-driven system that translates natural language descriptions into functional network topologies. The significance of this work lies in its potential to democratize network design education and streamline the network planning process. By bridging the gap between natural language and technical implementation, our solution makes network simulation more accessible to a broader audience, including students, educators, and professionals transitioning into networking.

II. Literature Review

A. Cisco Packet Tracer in Network Education

Packet Tracer has established itself as a fundamental tool in network education. According to Javid [1], it provides a virtual networking simulation environment that accurately represents real networking devices and their interactions. The platform offers both logical and physical workspaces, allowing users to design and test network configurations in a risk-free environment. Key features include real-time and simulation modes, which enable users to visualize packet flow and troubleshoot network issues effectively

B. AI in Network Management

Recent developments in artificial intelligence have significantly impacted network management and design. Mistry et al. [2] highlight how AI, particularly deep learning and reinforcement learning, has enhanced various aspects of networking, including routing optimization, traffic management, and security. Their research demonstrates AI's capability to improve network efficiency and reduce human intervention in network operations.

C. Natural Language Processing in Technical Domains

The application of natural language processing in technical domains has shown promising results in reducing complexity and improving accessibility. Large Language Models (LLMs) have demonstrated particular effectiveness in translating human intent into structured technical specifications, though their application in network design automation remains relatively unexplored.

III. Methodology

Our solution implements a multi-stage pipeline that transforms natural language inputs into functional Packet Tracer configurations:

A. File Format Analysis

- Reverse engineering of .pkt files revealed their XML-based structure.
- Identification of critical XML elements for device configurations and connections
- Development of encoding/decoding mechanisms for binary format conversion

B. Natural Language Processing Pipeline

- Primary LLM: Processes user input to extract network requirements
- Secondary LLM: Converts structured requirements into Packet Tracer-compatible XMLs
- Custom Python functions for XML manipulation and binary conversion

IV. Implementation Architecture

The process of automating network design and configuration begins with a natural language input provided by the user, describing the desired network setup. This input undergoes a series of transformative stages, leveraging the capabilities of advanced AI models and structured data processing techniques to produce a compilable ‘.pkt‘ file compatible with Cisco Packet Tracer. The pipeline is as follows:

A. User Input

The pipeline starts with a user-provided natural language description of the network. This input might include details about the devices involved, their configurations, interconnections, IP addressing schemes, and any specific requirements or constraints for the network design.

B. LLM1: Requirement Extraction

The first Large Language Model (LLM1) processes the user input to extract key requirements. Using its natural language understanding capabilities, it identifies and organizes essential components such as device types (e.g., routers, switches, end devices), their configurations, relationships, and connectivity requirements. The output of this step is a well-structured representation of the network in JSON format.

C. JSON Structure

The extracted requirements are represented as a JSON structure, which acts as a bridge between the natural language input and machine-readable configurations. This JSON format serves as a standardized way to describe network components, including their attributes, inter-device connections, and assigned IPs. It is both human-readable and machine-parsable, facilitating seamless transition to subsequent stages.

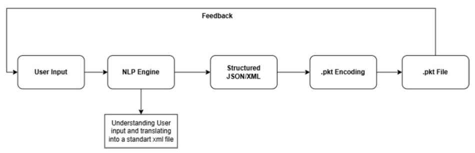


Fig. 1. Architecture for our project automation

D. LLM2: XML Generation

The second Large Language Model (LLM2) takes the JSON structure as input and converts it into XML content. XML is chosen for its hierarchical structure and compatibility with various network configuration tools. LLM2 ensures that the XML adheres to schema standards, accurately encoding the details of devices, their configurations, and interconnections. This step provides a compilable intermediate representation of the network.

E. Python Processing

The generated XML is processed using Python scripts. These scripts ensure correctness, validate the XML against predefined schemas, and prepare it for further encoding. Any inconsistencies or errors in the XML are flagged and corrected during this stage, ensuring the output aligns with the requirements specified by the user.

F. pkt File Generation

The final step involves encoding the XML into a ‘.pkt‘ file format, which is the standard used by Cisco Packet Tracer. This file serves as the complete representation of the network configuration, ready for simulation. The ‘.pkt‘ file includes all device settings, connections, and IP configurations, enabling users to load and test their network design in Cisco Packet Tracer seamlessly.

V. IV. Results and Discussion

The implementation demonstrates successful automation of network design through several key achievements

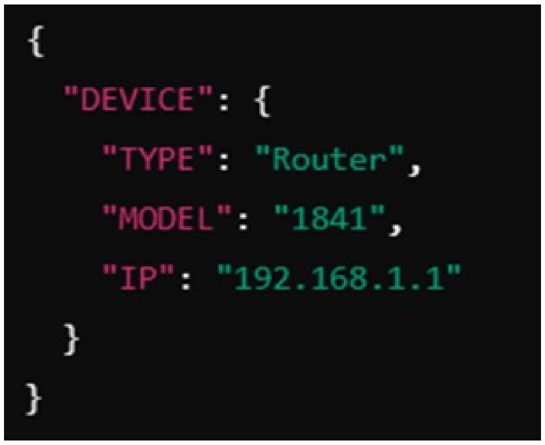

```
{
  "DEVICE": {
    "TYPE": "Router",
    "MODEL": "1841",
    "IP": "192.168.1.1"
  }
}
```

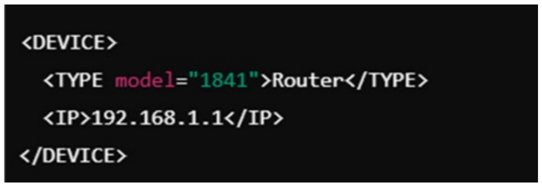
Fig. 2. User input is interpreted into devices and their configuration by LLM1

A. User input interpretation

When a user provides an input in natural language, the system utilizes a Large Language Model (LLM) (referred to as LLM 1) to process and extract specific components from the input. These components include device types and their configurations. For example, Fig. 2 illustrates the output for a router, where the extracted details include the router model and the assigned IP address.

B. Conversion into XML

After all components and their configurations have been identified, a second Large Language Model (LLM 2) processes this data and converts it into a structured XML format. The generated XML content is fully compilable and adheres to the schema required for further processing. This XML can subsequently be utilized to generate a .pkt file, which is compatible with network simulation tools. Fig 3 illustrates this converstion.

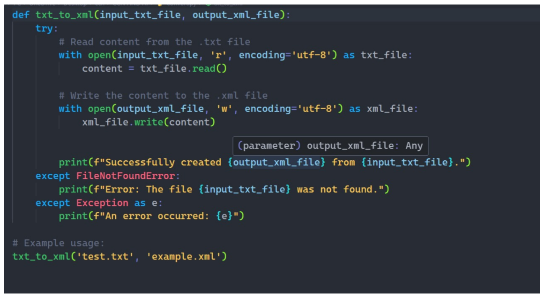


```
<DEVICE>
  <TYPE model="1841">Router</TYPE>
  <IP>192.168.1.1</IP>
</DEVICE>
```

Fig. 3. Components are converted into XML content

C. creation of XML file

Fig. 4 illustrates the function responsible for utilizing the XML content generated by LLM 2 to create a complete and valid XML file. This file is structured to be compilable, ensuring it meets the necessary requirements for subsequent conversion into a .pkt file. The .pkt file can then be used in network simulation and analysis tools, enabling practical implementation of the described configurations.



```
def txt_to_xml(input_txt_file, output_xml_file):
    try:
        # Read content from the .txt file
        with open(input_txt_file, 'r', encoding='utf-8') as txt_file:
            content = txt_file.read()

        # Write the content to the .xml file
        with open(output_xml_file, 'w', encoding='utf-8') as xml_file:
            xml_file.write(content)
                                        (parameter) output_xml_file: Any
        print(f"Successfully created {output_xml_file} from {input_txt_file}.")
    except FileNotFoundError:
        print(f"Error: The file {input_txt_file} was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")

# Example usage:
txt_to_xml('test.txt', 'example.xml')
```

Fig. 4. XML file creation

D. pkt encoding and Results in Cisco

Once the XML file is generated, it can be encoded into a .pkt file format, which is compatible with Cisco Packet Tracer Version 5. This encoding process ensures that the network configuration is seamlessly integrated and ready for simulation. Fig. 5 demonstrates the final output of the pipeline, showcasing the complete transformation from natural language input to a .pkt file usable in the Cisco Packet Tracer environment.

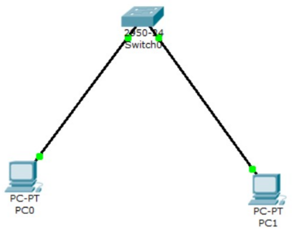


Fig. 5. Final packet tracer file. Containing one switch with 2 devices connected

VI. Key Findings

After extensive experimentation with the proposed approach and designing various architectures using Large Language Models (LLMs), our key findings are summarized below:

A. Natural Language Understanding

The system demonstrated robust capabilities in processing natural language inputs and accurately interpreting network configurations. Key aspects of its understanding include:

- Identification of network components and their interrelationships, enabling precise modeling.
- Recognition of connection requirements, ensuring appropriate network linkages.
- Parsing of IP addressing schemes, crucial for defining network interfaces.

B. Configuration Generation

The approach proved highly effective in generating network configurations with the following highlights:

- Accurate generation of XML schema, adhering to the required specifications.
- Automated device configuration, streamlining the setup process for network devices.
- Correct mapping of connections between devices, ensuring logical consistency.
- Efficient allocation of IP addresses based on the provided network description.

C. Limitations

Despite its strengths, the current implementation exhibits the following limitations:

- Compatibility is restricted to Cisco Packet Tracer Version 5, limiting broader applicability.
- Handling of complex network topologies remains a challenge and requires further refinement.
- Advanced networking protocols are currently unsupported, necessitating future development for comprehensive functionality.

VII. Conclusion and Future Work

This research successfully demonstrates the feasibility of AI-driven network design automation in Cisco Packet Tracer. The developed solution effectively reduces technical barriers while maintaining high accuracy in network configuration generation.

Future work should focus on the following areas:

- Extending compatibility to newer Packet Tracer versions, ensuring wider usability.
- Expanding support for complex network topologies, addressing scalability challenges.
- Implementing more sophisticated error handling to improve system robustness.
- Developing interactive feedback mechanisms to enhance user engagement and learning.

The findings from this research provide a strong foundation for further advancements in AI-assisted network design tools, particularly within educational and training contexts.

References

[1] S. R. Javid, ”Role of Packet Tracer in Learning Computer Networks,” *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 3, no. 5, pp. 6508–6511, May 2014.

[2] H. K. Mistry, C. Mavani, A. Goswami, and R. Patel, ”Artificial Intelligence for Networking,” *Educational Administration: Theory and Practice*, vol. 30, no. 7, pp. 813–821, 2024.

[3] T. P. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, ”Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[4] J. Mao, X. Li, Q. Zhang, and L. Ma, ”Neural architectures for named entity recognition,” *arXiv preprint arXiv:1603.01360*, 2016.