Updated Design Document

Project Architecture

This project implements a UDP-based file transfer system that is designed for efficiency and ease in data transmission. The system provides a user-friendly GUI for easy usage by users in order to send and receive files over the air with much ease.

Finite State Machine (FSM) Diagram - RDT 1.0 (Reliable Data Transfer Protocol 1.0)

The system is based on the RDT 1.0 model, which is the most basic form of a reliable data transfer protocol. It assumes a completely reliable channel; thus, there is no possibility of error, loss, or duplication during transmission.

Key Characteristics of RDT 1.0:

- The source continuously sends packets without any need for acknowledgements
- The receiver retrieves the incoming packets and forwards them to the application layer without further verification.
- Retransmissions, error detection, and acknowledgment mechanisms are not required because it is assumed that the communication channel is totally reliable.

FSM Representation of RDT 1.0

1. Sending Side (FSM - rdt1.0: sending side)

- The sender remains in "Wait for call from above" state; this means that it can accept new data.
- When data arrives from the application layer, the sender:
    1. Calls rdt_send(data) to start the transmission.
    2. Places the data in a packet using packet = make_pkt(data).
    3. Sends packet by calling udt_send(packet) - here, udt_send just means to send data over an unreliable data transfer layer but is assumed to be reliable in RDT 1.0.

2. Receiving Side (FSM - rdt1.0: receiving side)

- The receiver always stays in the "Wait for call from below" state, ready to accept new packets.
- Upon packet arrival, the receiver
    1. Calls rdt_rcv(packet) to receive it.

2. Extracts the data from the packet using extract(packet, data).

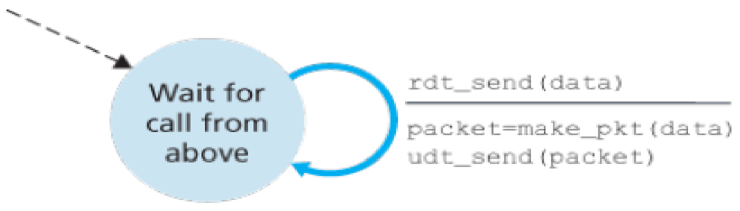
3. Delivers the data to the application layer using deliver_data(data), completing the transfer.

3.<u>Limitations of RDT 1.0</u>

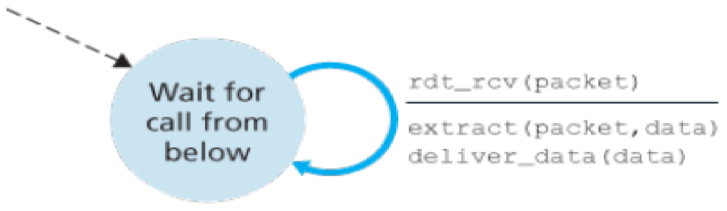
RDT 1.0 works fine over an error-free channel, but a real network doesn't guarantee that:

- Packet loss, corruption, and duplication may occur.
- Error detection and retransmission mechanisms are necessary for reliability.

For that reason, practical implementation employs higher versions of protocols like RDT 2.0 and RDT 3.0, which incorporate acknowledgments, error checking, and retransmissions.

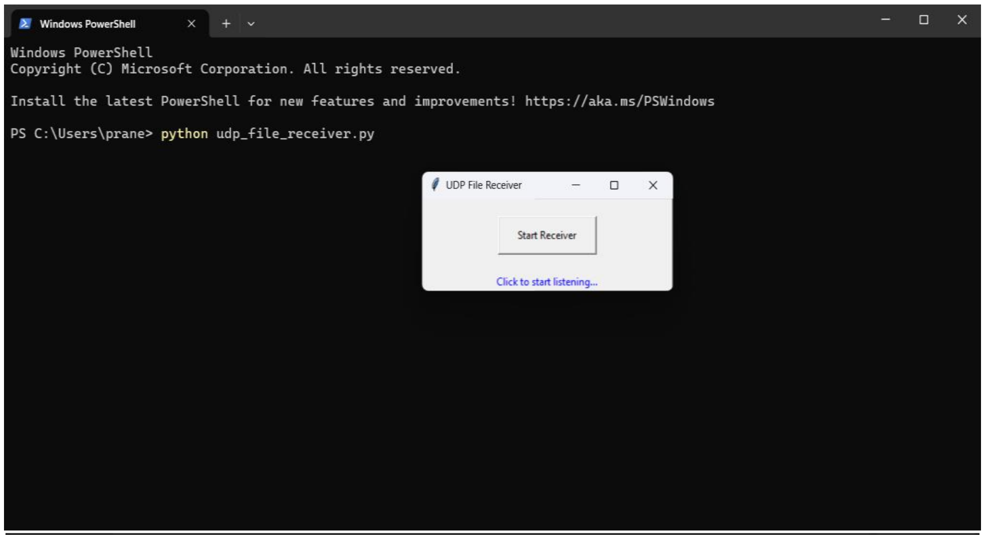


a. rdt1.0: sending side

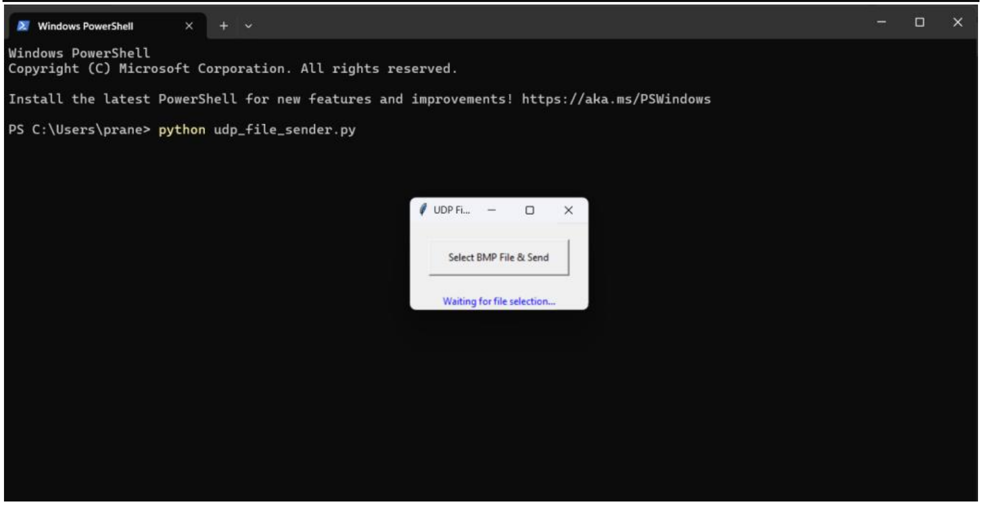


b. rdt1.0: receiving side

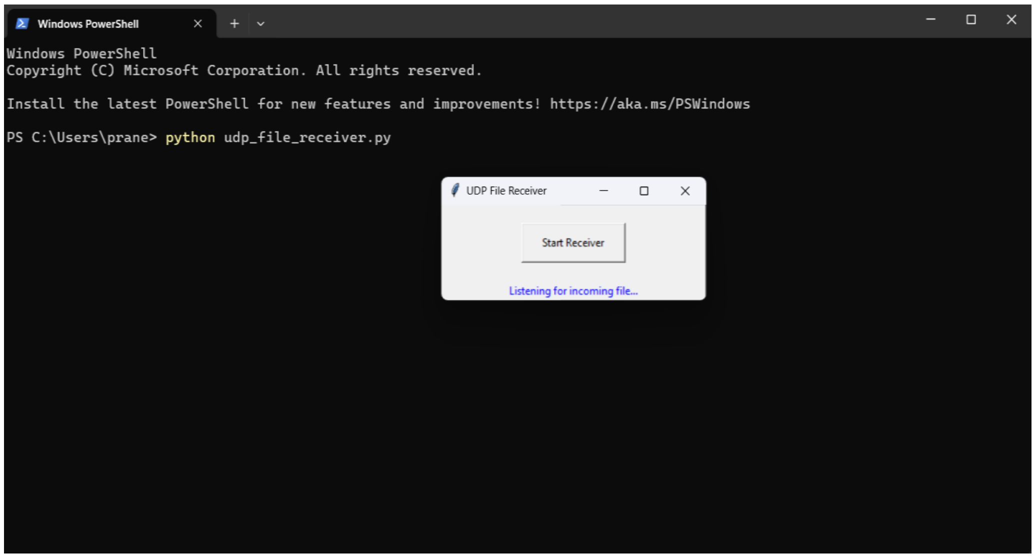
<u>Screenshots of Execution</u>



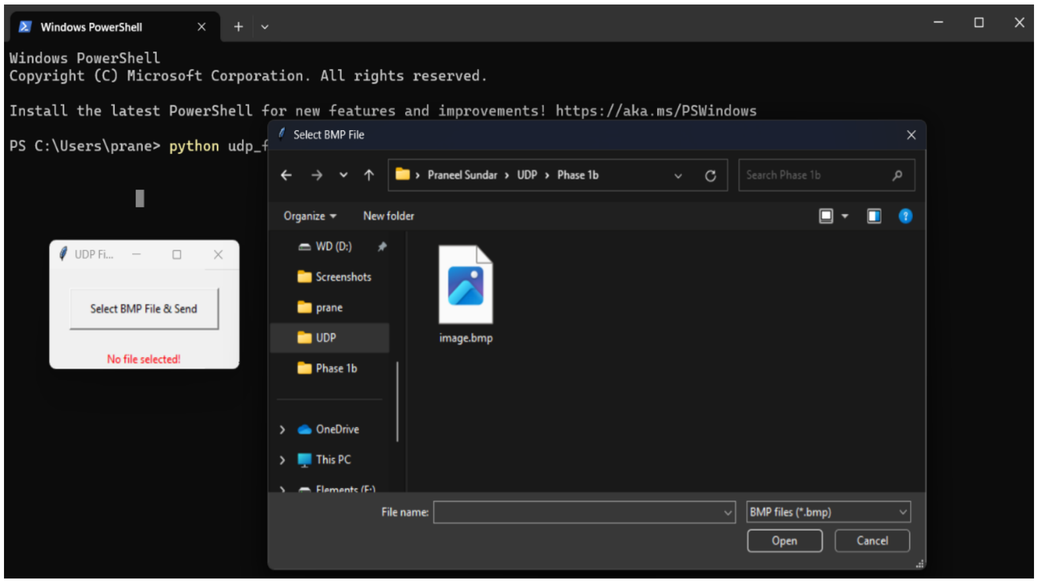
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\prane> python udp_file_receiver.py
```



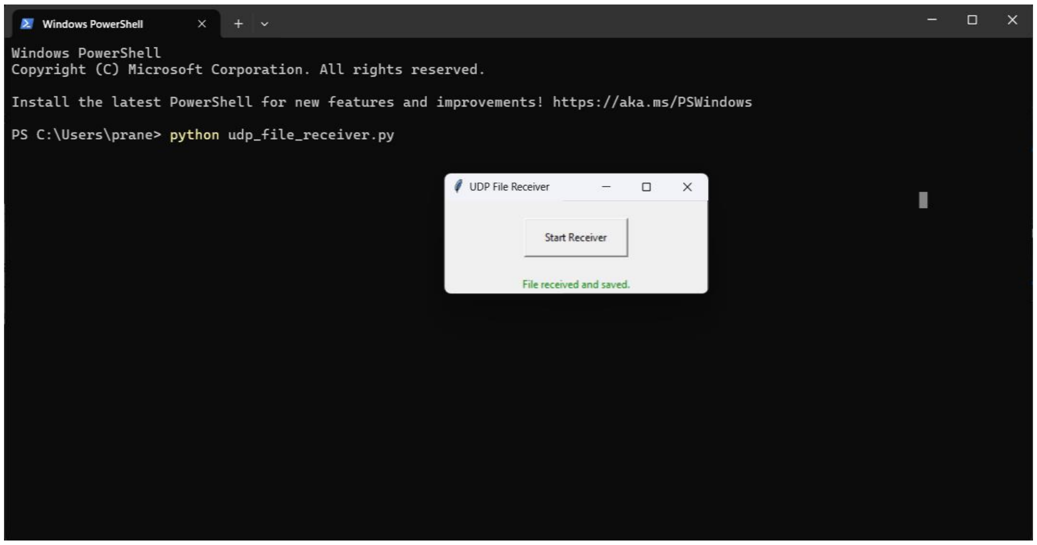
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\prane> python udp_file_sender.py
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\prane> python udp_file_receiver.py
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\prane> python udp_f
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\prane> python udp_file_receiver.py
```