Introduction to Computer Networks

Project phase 0

Remote File Storage using raspberry pi and IPFS

Abstract:

We normally store data on our laptops or other storage devices. These files eventually build up and take up space on our devices. Normally, files can be categorized into four types:

Regularly accessed: These files are used often and need to be stored locally.
Occasionally accessed: These files are used less often but are still important and need to be stored locally.
Important but once in a while accessed: These files are important but are only accessed occasionally. They can be stored in a more remote location, such as a cloud storage service.
Never touch it again: These files are no longer needed and can be deleted.
We are solving the problem of storing important but once in a while accessed files by using a Raspberry Pi as a server and IPFS as file storage.

What is IPFS?

IPFS is a peer-to-peer file system that uses content-addressing to uniquely identify files. It is more resilient, accessible, and less centralized than traditional file-sharing systems.

Here are some of the benefits of IPFS:

Resilience: IPFS is a distributed system, so if one node goes down, the data is still available on other nodes.
Accessibility: IPFS can be accessed from anywhere in the world, as long as there is an IPFS node available.
Decentralization: IPFS does not rely on a central server, so it is not subject to the control of any single entity.
What do we do using Raspberry Pi?

We initialize an IPFS node on the Raspberry Pi so that we can upload and download files from the decentralized storage system. We will provide two options at the start: one is to create an account and the other is to log in. Once a user creates an account, they can log in to the server. Once they log in, they can upload files to the server from their PC using different protocols. Once the file is received by the server, it will check from which user the file was received. Based on that, the file will be transferred to that particular user's folder, which was created at the time of account creation. Here is where IPFS comes in. Once the file is uploaded to the user's folder, our server will upload that file to IPFS and we will get a Content ID as output. We will then store the Content ID and File name as a key-value pair in the SQLite database under the user table.

If the user wants the file to be downloaded, they can simply type in the file name in the server and the server will download the file from IPFS and send it to the user's local machine using different protocols based on the type of file.

Note: Once the file is uploaded to IPFS, the server will no longer have the file. The server will delete the file from the server's local file system, but IPFS will have a copy of the file. This saves storage space on the server.

This Raspberry Pi can handle requests from many users at a time and can be accessed by any network from any part of the world. We are using tunneling from third-party services for this because Amrita Connect does not support port forwarding.

Every function will be working concurrently, which will lead to concurrent execution. This will allow us to accommodate many more users who can access the system quickly without having to wait for other users to complete their requests.

We are using Rust to solve the slow execution problem that arises from Python.