Kyle Pellerin and Mati Gibbons Present...

A Moderately Centralized, Safety-Oriented, Peer to Peer File Transfer System

AGENDA

01 The Vision

02 Architecture Overview

03 Design and Components

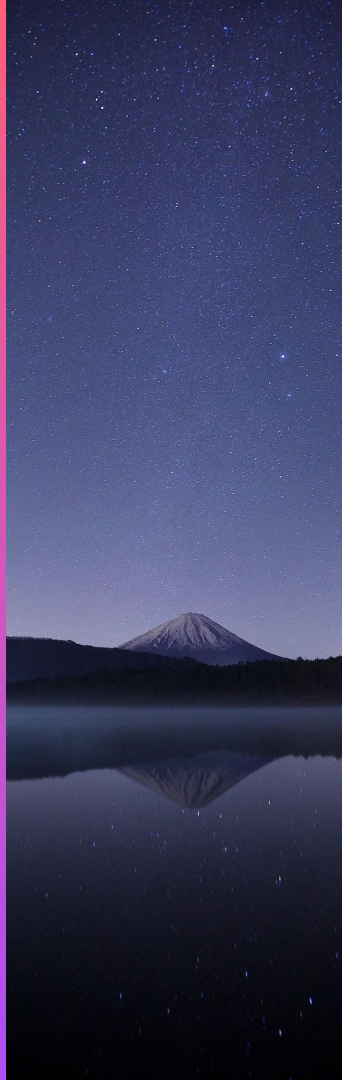
04 Implementation

05 Expected Evaluation

06 Future Steps

THE VISION

- Create a P2P file transfer system capable of near linear scaling
- Reduce single points of failure in the centralized components
- Provide an accessible interface and easy connection to users
- Prioritize the safety and anonymity of the system



Why Do our goals Matter?

01

Linear Scalability

- The more people that have the file, the faster it transfers
- Performance (latency) is a key metric in any Distributed System

02

Single Points of Failure

- Reduce down time and keep service at highest uptime possible
- Benefits of centralized architecture without the stress of SPOOFs

03

Ease of Use

- The more users online, the more files available and the faster file transfers will be

04

Safety and Anonymity

- Control of the system is essential, keep users safe from viruses and potential illicit material
- Users should be anonymous to protect their identity, and have the resource to report malicious use

The Architecture

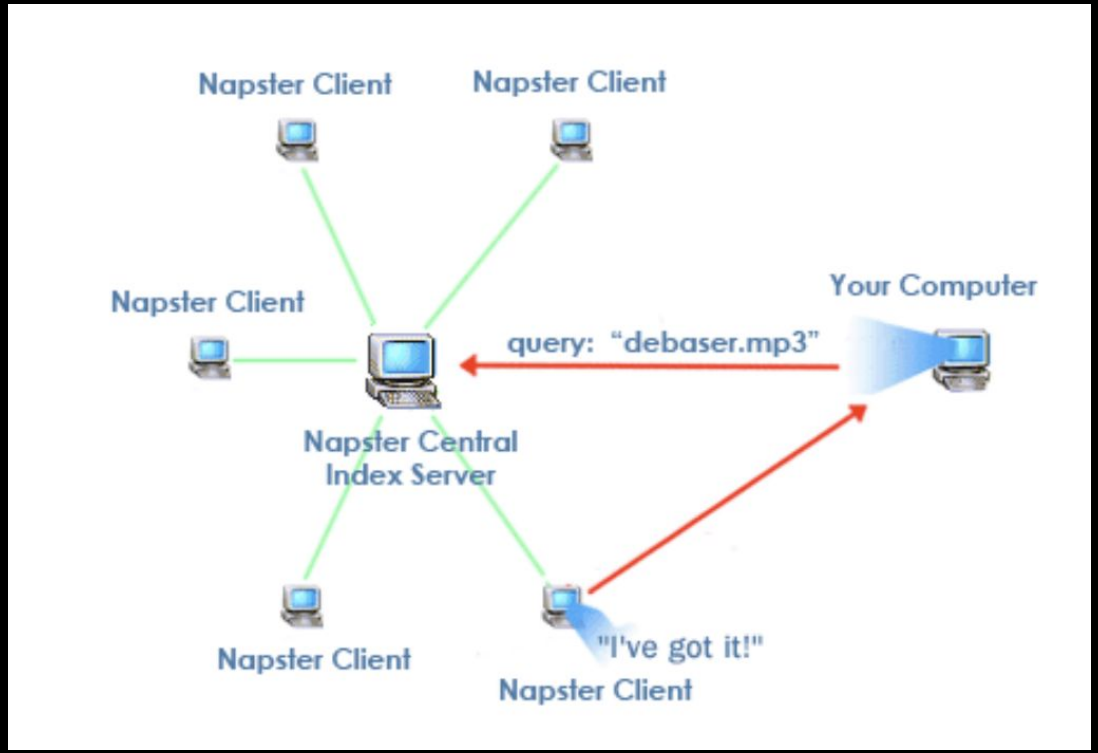
2. The Peer

1. The Server

3. FileList

The Server

- Built in Java on XML RPC
- Main and Backup and run through one file that initializes them both
- Initialize data structures on both
- Users can register when joining the network via a client
- Search through files, organize downloads, peer conducts the download



The Peer

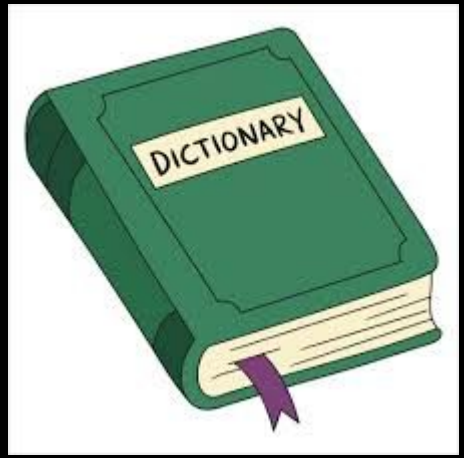
- Use XML to connect to server via IP
- Contains all the User interface code
- Uses Flask to run its own server for P2P File Share
- Users can search, request to download, listen, or quit
- Make as simple and user friendly as possible

```
Welcome to the P2P File Sharing Network
Please offer the main server what files you can share:
Enter filenames separated by commas: file1.txt, secrets.txt, bruh.png
You are sharing the following files: ['file1.txt', 'secrets.txt', 'bruh.png']
127.0.0.1 - - [08/Dec/2025 19:32:34] "POST /RPC2 HTTP/1.1" 404 -
Error registering files: <ProtocolError for localhost:8089/RPC2: 404 NOT FOUND>

--- MENU ---
1. SEARCH for a file
2. DOWNLOAD a file
3. Listen quietly for incoming requests
4. Exit
Selection:
```

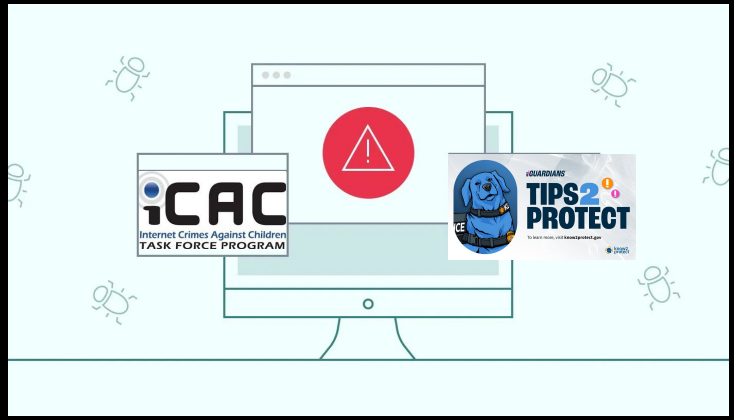
FileList

- Store what IP has what file
- File list fragments are parsed when a peer joins the network
- File lists are updated on successful transfer
- Writes to the Main server's FileList and written to the Backup's
- Consistency between the Main and backup servers file list is design to be nearly instantaneous, low traffic across it



Safety Through Reporting

- Many P2P systems struggle from malicious intent from users.
- Allow users after a file share to report malicious content shared.
- Then on our centralized server system we will hold a list of blacklisted users.
- Blocking these users from connecting and sharing content.
- Time permitting we could use an algorithm to scan for malicious file names

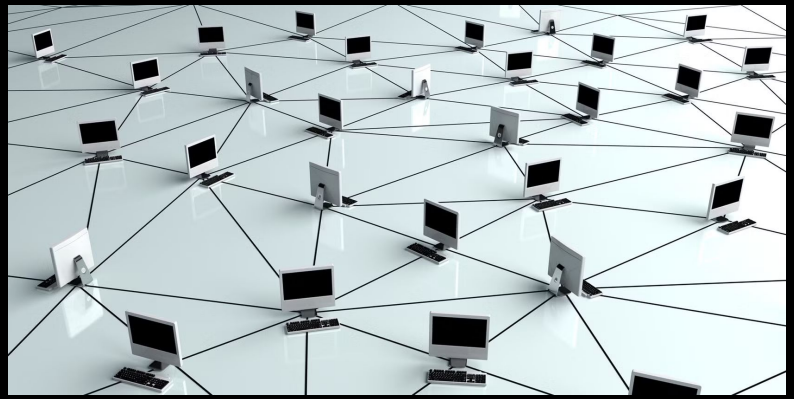


Peer-to-Peer File Sharing Case Law Review

By Dennis Nicewander
Assistant State Attorney
Broward County, Florida

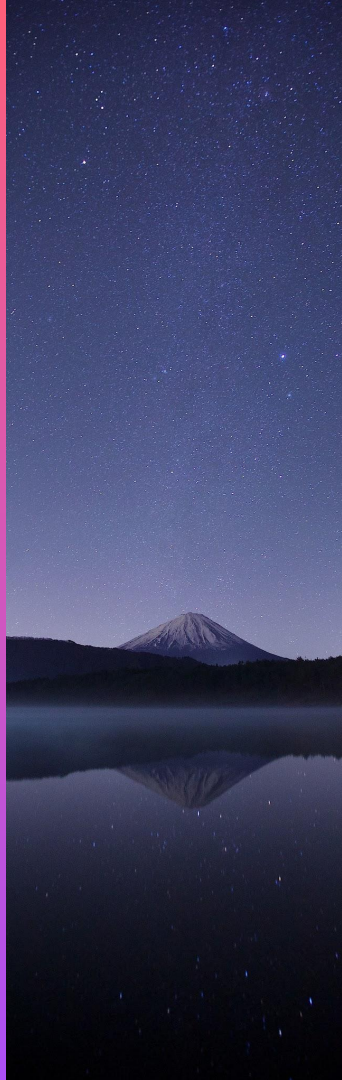
Multi Peer Sharing

- We plan to implement our system to allow multiple peers to share a file at once.
- Distributing the load across multiple clients
- We expect this to also decrease share time.
- Also providing safety for sharing should a peer fail in the process of sharing.



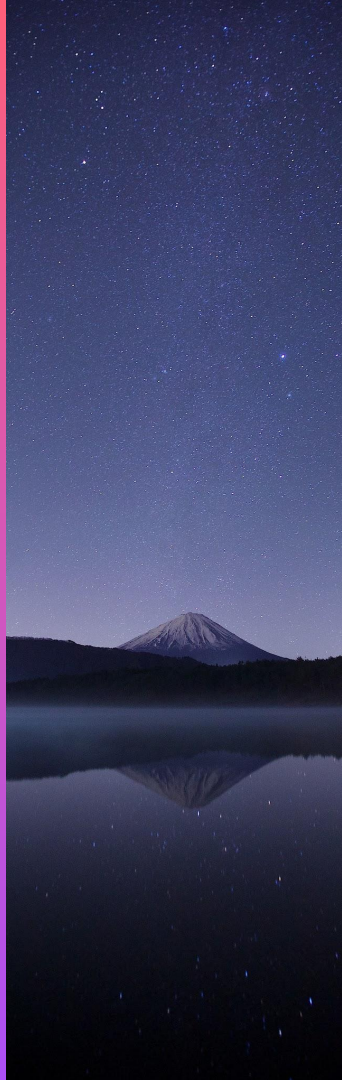
Current Implementation Status

- Supports single P2P File Transfers (assuming all peers who have a file have the whole file)
- Backup automatically takes over workload when main fails. As soon as main recovers it takes over the process.
- Works Locally and globally across servers given in class
- Reporting and overwriting prompts are available but need to be set up



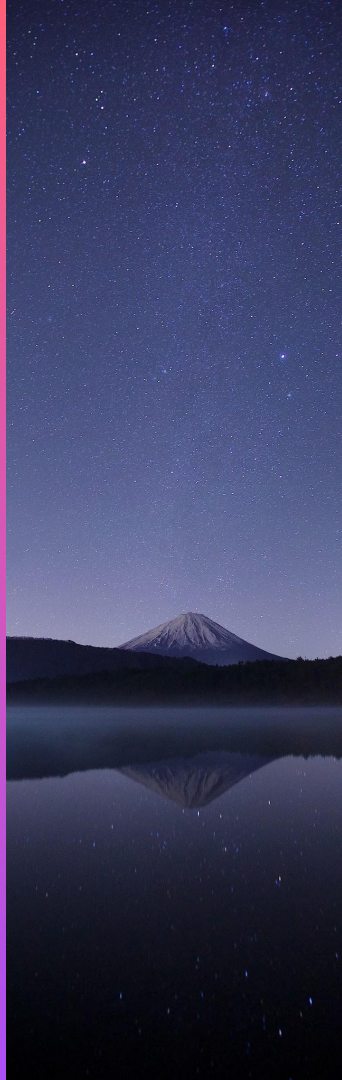
Expected Evaluation

- Evaluate file transfer speed
  - Number of requests
  - Number of peers with file
  - Failure of master server
- Increase or decrease in speed depending on number of peers sharing.
  - Across large files (books) and small files (low resolution images)
- Server load as more peers connect at once.



Future Steps

- Move peer file selection from peer to invisible to preserve anonymity (currently useful for debugging)
- Enable multiple peers to transfer parts of a file to single requestor
- Enable file overwrite as an option instead of a default
- Potentially change backup client connection to be static instead of a dynamic check
- Check restart status of main server



THANK YOU

Any questions?