Distributed Systems

Project Proposal
Gnutella: Consistency
Team 32

Kritin Maddireddy (2022101071)
Kandi Jayanth Reddy (2022101038)

November 20, 2024

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

1 Problem Statement

The Gnutella protocol, a decentralized peer-to-peer (P2P) file-sharing protocol, relies on distributed servants (peer entities functioning as both client and server). This approach allows for redundancy, scalability, and fault tolerance. However, the lack of global consistency mechanisms introduces challenges for data synchronization and accuracy across peers, leading to problems like stale or incomplete search results, inefficiencies, and the risk of data discrepancies. Our project aims to address these consistency issues by introducing mechanisms to synchronize state and data consistency across servants in the Gnutella network.

2 Material

We plan to read and implement the features of the following papers.

- **Gnutella Protocol** v0.4, v0.6 **Specification**: Provides the current protocol design, including the descriptor-based communication model and limitations for consistency in data propagation and synchronization
- Peer-to-Peer Systems and Applications: Covers foundational P2P concepts, including scalability, fault tolerance, and various consistency mechanisms.
- **Conflict-free Replicated Data Type, or CRDT (potential):** These are explored as potential models to adapt for consistency across the Gnutella network, focusing on decentralized, peer-to-peer synchronization without central coordination.

3 Scope of the project

This project proposes to integrate a consistency layer within the Gnutella protocol. Specifically, we plan to introduce a distributed consistency mechanism that ensures:

- **Data Synchronization**: Ensuring that file availability data and search indices are consistent across peers to avoid stale search results.
- **State Consistency**: Allowing servants to maintain a consistent view of network participants and file resources without reliance on a central server.
- **Optimized Traffic Management**: Reducing redundant queries and pings by introducing more efficient routing algorithms that consider consistency checks.

Use Cases Addressed:

- **Improved Search Accuracy**: By synchronizing indices across servants, users experience fewer outdated or incomplete search results.
- **Reduced Redundancy**: By maintaining state consistency, redundant queries and update pings are minimized, leading to optimized network traffic and lower resource consumption.

4 Implementation approach

- ***Technologies:***
  - **Programming Language:** Rust for core components (and backend)
  - **Version control:** Git
  - **Frontend UI/UX:** NextJS with Typescript (optional, depending on how much time we have left)

Our solution involves incorporating distributed consistency mechanisms into Gnutella's existing descriptor framework (Ping, Pong, Query, QueryHit, and Push). We will implement this in two main parts:

- **State Synchronization Mechanism**: Using a modified version of Conflict-free Replicated Data Types (CRDTs), each servant maintains a partially ordered set of known files and peer availability status. Updates are broadcast selectively, optimizing network resources by avoiding redundant updates. CRDTs will help manage synchronization in cases of peer failure or rejoining, ensuring consistency without direct dependencies on specific peers.
- **Version Control for Data Consistency**: For query and file-sharing consistency, a version control layer will tag and track file versions across peers, ensuring accurate responses to file requests. This layer will use a timestamp or vector-based protocol to resolve conflicts when multiple versions of a file are present.

Note: These are the possible implementation routes we are thinking of currently, and we may extend/modify/discard these altogether while implementing, depending on the practicality and feasibility of said implementation.

5 Timelines (tentative)

Total time: 4 weeks

- **Week-1 and Week-2:** Implement the existing Gnutella protocol and core functionality, and start researching consistency on the Gnutella protocol.
- **Week-3:** Research and implement consistency on the Gnutella protocol.
- **Week-4:** Bug testing and fixing, and if time permits, optionally design and implement the frontend UI.