INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

PROJECT REPORT
OF
GNUTELLA - HETEROGENEITY

Team Name: Team 13

Team Members: Lakhsya Namwani (2021701002)

Harshita Harshita (2023201002)

Course: Distributed Systems

Session: Monsoon 2024

Gnutella with Heterogeneity

Introduction

Peer-to-peer (P2P) networks represent a significant shift from traditional client-server models by enabling decentralized communication and resource sharing among nodes (peers). In a P2P network, nodes collaborate directly to distribute workload, store data, and facilitate efficient communication without relying on a central server. Such networks are highly resilient, scalable, and capable of operating even under partial network failures.

This project aims to replicate the core functionalities of the **Gnutella protocol**, a pioneering decentralized file-sharing system, by implementing mechanisms for **node discovery, connectivity maintenance, communication**, and **file transfer**. The Gnutella protocol is renowned for its robustness and decentralized architecture, which ensures that no single point of failure can disrupt the network.

The system is divided into two primary components:

1. **Bootstrap Server**: A temporary central entity that assists new nodes in joining the network by providing information about existing peers.
2. **Peer Nodes**: Autonomous entities that participate in the network by discovering other peers, searching for files, and sharing resources.

By leveraging controlled flooding for peer discovery and file searches, this project emphasizes dynamic connectivity, redundancy, and efficient data exchange. The implementation not only simulates a Gnutella-like network but also showcases essential principles of distributed systems, such as fault tolerance and scalability.

Components of the System

1. Bootstrap Server

The bootstrap server is a centralized entity used only at the initial stage to help new nodes join the network. It maintains a list of active nodes with their respective IP addresses and ports. When a node connects to the bootstrap server, it retrieves the address of a random peer already in the network.

Key Responsibilities:

- Maintain a dynamic list of active peers.
- Provide random peer information to newly joining nodes.
- Facilitate the creation of a connected graph of nodes.

2. Peer Nodes

Nodes are the core of the P2P network, performing the following tasks:

- Connecting to other peers based on the information provided by the bootstrap server.
- Sending and receiving **PING** and **PONG** messages to establish and strengthen connectivity.
- Searching for files using **QUERY** messages and responding with **QUERYHIT** messages.
- Handling file transfer based on bandwidth to ensure efficient data exchange.

Functionalities and Workflow

1. Node Initialization

1. The bootstrap server must be started first to ensure the network can handle joining requests.
2. A node starts by executing nodes.py, which:
    - Connects to the bootstrap server to obtain the IP address and port of a random peer.
    - Establishes a connection with the peer retrieved from the bootstrap server.

2. Node Connectivity

After establishing an initial connection, the node performs the following:

- **PING Flooding:**
    - The node sends a **PING** message with a limited TTL (Time-to-Live) to discover other peers in the network.
    - When a peer receives a PING, it responds with a **PONG** message, which traces back the path of the PING.
    - This mechanism ensures a robust and interconnected network.
- **PONG Responses:**
    - PONG messages carry information about the responding node, including IP, port, and available bandwidth.
    - These responses help maintain a connected graph of nodes, ensuring redundancy and strong connectivity.

3. File Search

Nodes can request files by flooding the network with a **QUERY** message:

- The QUERY is propagated similarly to PING, with a limited TTL to prevent excessive flooding.
- Nodes that possess the requested file respond with a **QUERYHIT** message.
- QUERYHIT messages trace the same path as the QUERY, ensuring that the requesting node receives all potential matches.

4. File Transfer

Once the requesting node receives multiple QUERYHIT responses, it evaluates the available options:

- The node selects the peer with the **highest bandwidth** to ensure efficient file transfer.
- The file is then transferred from the selected peer to the requesting node.

Message Types

1. PING

- **Purpose:** To discover new peers in the network.
- **Propagation:** Flooded to neighboring nodes with a TTL value.
- **Response:** Generates a PONG message.

2. PONG

- **Purpose:** To confirm connectivity and provide information about the responding peer.
- **Propagation:** Traces the path of the originating PING.

3. QUERY

- **Purpose:** To search for a specific file in the network.
- **Propagation:** Flooded with a TTL value.
- **Response:** Generates QUERYHIT messages from peers that have the requested file.

4. QUERYHIT

- **Purpose:** To confirm the availability of the requested file and provide details for retrieval.
- **Propagation:** Traces the path of the originating QUERY.

Key Features

Decentralized Architecture

The system does not rely on a central server for file storage or retrieval. Each peer contributes to the network, making it resilient to single points of failure.

Flooding Mechanism

Messages like PING and QUERY are propagated using controlled flooding to ensure all nodes are reachable without overwhelming the network.

Selective File Transfer

By choosing the peer with the highest bandwidth for file transfer, the system optimizes resource usage and minimizes transfer time.

Dynamic Peer Discovery

Nodes dynamically discover and maintain connections with other peers, ensuring the network remains robust and functional even as peers join or leave.

Implementation Details

Technologies Used

- **Programming Language:** Python
- **Networking:** Python's socket library for communication between nodes and the bootstrap server.

Key Files

1. **bootstrap.py** - Implements the bootstrap server.
2. **nodes.py** - Handles the main functionality of peer nodes, including connectivity, message exchange, and file transfer.

Challenges and Solutions

1. Preventing Infinite Flooding

- **Challenge:** Uncontrolled flooding of PING and QUERY messages could lead to excessive network traffic.
- **Solution:** Introduced a TTL (Time-to-Live) value for messages to limit their propagation.

2. Maintaining Connectivity

- **Challenge:** Ensuring that nodes remain connected despite peers joining and leaving the network.
- **Solution:** Regular PING exchanges allow nodes to maintain an updated view of the network.

3. Efficient File Transfer

- **Challenge:** Optimizing file transfer to minimize time and bandwidth usage.
- **Solution:** Nodes select the peer with the highest bandwidth for file transfer, ensuring efficient resource utilization.

Future Enhancements and Conclusion

1. Enhanced Query Mechanisms

Implement advanced search techniques like indexing or bloom filters to reduce QUERY flooding.

2. Secure Communication

Integrate encryption to ensure secure message exchanges and file transfers.

3. Fault Tolerance

Introduce mechanisms to handle network partitioning and node failures more gracefully.

We successfully implemented the core functionalities of the Gnutella protocol, including dynamic peer discovery, robust connectivity, and efficient file sharing. By combining decentralized architecture with controlled flooding, the system ensures scalability and resilience, making it an effective model for P2P networks.