Bootstrap 🤝

The application serves as a bootstrap node, providing an HTTP endpoint to distribute known peers to new peers joining the network. This setup ensures seamless peer discovery and initialization, enabling efficient and secure distributed data exchange within the peer network. By containerizing the application using Docker, deployment becomes significantly easier and makes quick VPS deployments easy. Additionally, the user control panel offers finer-grained controls over the network, including scaling and monitoring, which enhances the manageability and reliability of the peer network infrastructure.

Initialization

The `P2PBootstrap` project initializes by setting up the necessary configurations and services required for the bootstrap server to operate. The main entry point is the `Program.cs` file, which configures the application and starts the web server.

1. **Configuration**: The application reads configuration settings from the `appsettings.json` file. This includes settings for encryption keys, database paths, and other essential configurations.
2. **Logging**: Logging is configured to use a plain text format and is activated to capture important events and errors.
3. **Web Server Setup**: The application uses ASP.NET Core to set up the web server. It configures the HTTP request pipeline, enabling default files, static files, and routing. This is an AOT compatible application, as is most of the P2PNet library.

Operation

The `P2PBootstrap` project operates by providing several key functionalities:

1. **Peer Distribution**: The bootstrap server provides an HTTP endpoint (`/api/Bootstrap/peers`) to distribute the list of known peers to new peers joining the network. This endpoint can operate in two modes:
    - **Trustless Mode**: Returns the list of known peers directly.
    - **Authority Mode**: Requires the client to receive and store a public key from the bootstrap server before returning the peer list.
2. **Parser Integration**: The server integrates with a parser to handle input and output operations. It provides endpoints to get parser output (`/api/parser/output`) and to submit parser input (`/api/parser/input`).
3. **Encryption Service**: The server initializes an encryption service to handle secure communication. This includes generating and loading PGP keys from the specified directory.

4. **Database Initialization**: The server initializes a local database to store necessary data. It ensures the database directory exists and sets up the required files.

User Control Panel

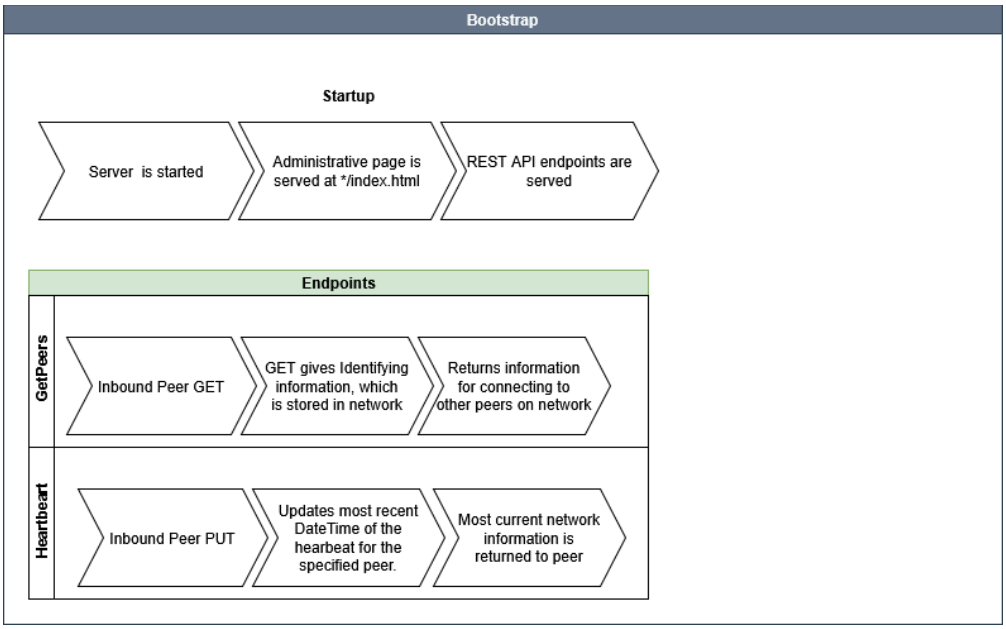
The user control panel provides a web-based interface for managing the bootstrap server. In this web-based interface is a terminal for executing commands. It includes the following pages for easier management as well:

1. **Overview**: Displays an overview of the server's status and key metrics.
2. **Settings**: Allows users to modify server settings, such as encryption keys and database paths.
3. **Peers**: Displays the list of known peers and provides options to perform actions on them, such as disconnecting or blocking peers.

Diagrams

To supplement the information visually, the following diagrams are provided:

1. **Bootstrap Server Architecture**: Shows the overall architecture of the bootstrap server, including its interaction with the P2P network and the user control panel.
2. **Peer Distribution Flow**: Illustrates the flow of peer distribution, from a new peer requesting the peer list to the server returning the list based on the configured trust policy.



Note: Bootstrap server still under construction

Widescan

The `P2PNet.Widescan` class library project is designed to facilitate mass IPv6 address generation and peer discovery within a peer-to-peer network. This project leverages hardware capabilities, such as GPUs, to efficiently generate vast quantities of IPv6 addresses. By utilizing user-defined address prefixes, it allows for a more targeted and narrow scope of addresses to ping, enhancing the efficiency of the discovery process. This can be leveraged with publicly available information on IPv6 prefix registrations, like the Ripe database, in order to refine the scope of the widescan. Additionally, the project employs lightweight ICMP packets to broadcast discovery information, ensuring minimal network overhead while effectively communicating with potential peers.

Initialization

The `P2PNet.Widescan` project initializes by setting up the necessary configurations and services required for widescan operations. The main entry point is the `Widescan` class, which configures the application and starts the widescan process.

1. **Configuration**: The application reads configuration settings from the provided parameters. This includes settings for address prefixes, hardware mode (GPU or CPU), and other essential configurations.
2. **Logging**: Logging is configured to use a plain text format and is activated to capture important events and errors.
3. **Hardware Mode Setup**: The application can be configured to use either GPU offloading or parallel CPU capabilities for address generation and peer discovery.

Operation

The `P2PNet.Widescan` project operates by providing several key functionalities:

1. **IPv6 Address Generation**: The widescan application generates vast quantities of IPv6 addresses using either GPU offloading or parallel CPU capabilities. This allows for efficient and rapid address generation.
    - **GPU Offloading**: Utilizes the processing power of GPUs to generate IPv6 addresses in parallel, significantly speeding up the process.
    - **Parallel CPU Capabilities**: Uses multiple CPU cores to generate IPv6 addresses in parallel, providing an alternative for systems without GPU support.
2. **Peer Discovery**: The widescan application pings the generated IPv6 addresses to discover potential peers within the network. It uses lightweight ICMP packets to broadcast discovery information, ensuring minimal network overhead.

3. **Address Prefix Filtering**: By utilizing user-defined address prefixes, the widescan application narrows the scope of addresses to ping, enhancing the efficiency of the discovery process. This can be refined using publicly available information on IPv6 prefix registrations, such as the Ripe database.

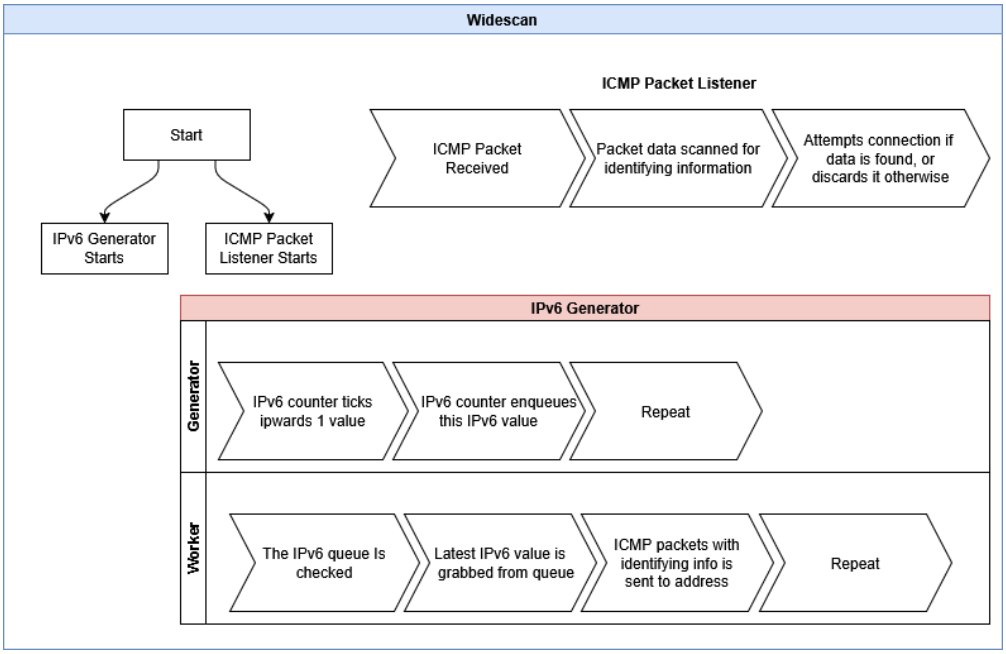
Integration

The `P2PNet.Widescan` project is designed to be a modular import that can be integrated with other P2P network applications, such as a`P2PNetwork` client application or the `P2PBootstrap` server. It does not run independently and does not use `appsettings.json` for configuration. Instead, it relies on the host application to provide the necessary configuration parameters.

Diagrams

To supplement the information visually, the following diagrams are provided:

1. **Widescan Architecture**: Shows the overall architecture of the widescan application, including its interaction with the P2P network and the hardware components (GPU/CPU).
2. **IPv6 Address Generation Flow**: Illustrates the flow of IPv6 address generation, from configuration to address generation using GPU or CPU, and finally to peer discovery. This is very much a simple consumer-producer pattern, with intermediate in-memory queues to operate as a broadcaster.
3. **ICMP Packet Listener:** Operating independently of the broadcaster, the ICMP Packet Listener operates as a listener for responses from prospective peers and other potential widescan instances. This utilizes a docile form of packet sniffing on the host machine.



Note: Widescan project still under construction 🏗️