canip_peer v4.1 — Usage Manual

Overview

`canip_peer` is a lightweight Python tool for bridging CAN bus traffic over TCP/IP networks. It enables seamless forwarding of CAN frames between devices such as Raspberry Pi with CAN-USB adapters and Linux PCs running SocketCAN. This makes it possible to monitor, debug, or extend CAN networks remotely over Ethernet, Wi-Fi, or even VPNs — without expensive hardware gateways.

Features

- Linux native (SocketCAN) — tested on Mint and Raspberry Pi OS (Raspbian)
- CAN ↔ TCP bridging — forward CAN frames to/from a TCP client/server
- Heartbeat support — detects broken links and idle peers
- Flexible filtering — pass/block CAN IDs via lists, ranges, or masks
- Sequencing option — detect missing/duplicate frames across the tunnel
- Logging — log CAN frames with timestamps for later analysis
- Lightweight — pure Python 3, no kernel modules required
- Testable with VCAN — works with Linux's virtual CAN devices for safe testing

Installation

Clone the repository and install dependencies directly (no requirements.txt needed):

```
sudo apt update
sudo apt install python3 python3-pip can-utils
git clone https://github.com/mackelec/CAN_IP.git
cd CAN_IP
pip3 install python-can
# Optional: pip3 install colorlog
```

Preparing CAN Devices

Enable CAN-USB (CandleLight/gs_usb):

```
sudo ip link set can0 up type can bitrate 500000
ip -details link show can0
```

Create a Virtual CAN device (for testing):

```
sudo modprobe vcan
sudo ip link add dev vcan0 type vcan
sudo ip link set vcan0 up
```

Usage

Run `canip_peer.py` in server mode on the Raspberry Pi:

```
python3 canip_peer.py server --can can0 --port 3333 --log-level INFO
```

On the Linux PC, run in client mode:

```
sudo ip link add dev vcan0 type vcan
sudo ip link set vcan0 up

python3 canip_peer.py client --can vcan0 --host <pi-ip> --port 3333 --heartbeat 2 --log-level INFO
```

Examples

1. Test with VCAN only (both ends on one machine)

```
python3 canip_peer.py server --can vcan0 --port 3333
python3 canip_peer.py client --can vcan1 --host 127.0.0.1 --port 3333
```

```
cansend vcan0 123#11223344
candump vcan1
```

2. RPi (with CAN-USB) to Linux PC

```
sudo ip link set can0 up type can bitrate 500000
python3 canip_peer.py server --can can0 --port 3333 --log-level INFO

sudo ip link add dev vcan0 type vcan
sudo ip link set vcan0 up
python3 canip_peer.py client --can vcan0 --host <pi-ip> --port 3333 --heartbeat 2 --log-level INFO

candump vcan0
```

Options

- --can : CAN device name (e.g. can0, vcan0)
- --port : TCP port (default 3333)
- --host : Server IP (for client mode)
- --heartbeat : Send heartbeat messages every N seconds
- --log-level : DEBUG, INFO, WARNING, ERROR
- --no-seq : Disable sequence checking

Troubleshooting

- CAN device not found: Check with `ip link show`
- No traffic on client: Verify firewall allows TCP 3333
- Continuous traffic loop: Ensure you don’t connect both sides to the same CAN bus simultaneously