Modules

my_ping

Minimal starter for a raw-socket ICMP ping tool.

my_ping.**build_icmp_packet**(*identifier: int*, *sequence: int*, *payload_size: int*) → bytes [source]

Build one ICMP echo request packet.

my_ping.**checksum**(*data: bytes*) → int [source]

Compute Internet checksum for ICMP header+payload.

my_ping.**main**() → int ¶ [source]

my_ping.**parse_args**() → Namespace [source]

my_ping.**send_ping**(*sock: socket*, *target: str*, *identifier: int*, *sequence: int*, *size: int*) → float [source]

my_traceroute

my_traceroute.py - Custom traceroute implementation using raw sockets.

Sends UDP packets with incrementing TTL values and listens for ICMP time-exceeded and port-unreachable messages to discover the route.

Usage:
sudo python my_traceroute.py [-n] [-q nqueries] [-S] <destination>

Requires root/administrator privileges to use raw sockets.

my_traceroute.**checksum**(*data: bytes*) → int [source]

Compute the Internet checksum (RFC 1071) over *data*.

Addition, if carry then wrap around then do 1's complement

Parameters: **data** – Raw bytes to checksum.
Returns: 16-bit checksum as an integer.

my_traceroute.**create_recv_socket**(*timeout: float*) → socket [source]

Create a raw ICMP socket for receiving error messages.

Parameters: **timeout** – Socket receive timeout in seconds.
Returns: Configured raw ICMP socket.

my_traceroute.**create_send_socket**(*ttl: int*) → socket [source]

Create a UDP socket with the given TTL.

Parameters: **ttl** – Time-to-live value for outgoing packets.
Returns: Configured UDP socket.

my_traceroute.**main**() → None [source]

Entry point for my_traceroute.

my_traceroute.**parse_args**() → Namespace [source]

Parse command-line arguments.

Returns: Parsed argument namespace.

my_traceroute.**resolve_hostname**(*addr: str*) → str [source]

Attempt a reverse DNS lookup for *addr*.

Parameters: **addr** – IP address string to look up.
Returns: Hostname string, or the original *addr* if lookup fails.

my_traceroute.**send_probe**(*recv_sock: socket*, *dest_addr: str*, *ttl: int*, *port: int*) → tuple[str | None, float | None] [source]

Send a single UDP probe and wait for an ICMP response.

Parameters:
- **recv_sock** – Raw ICMP socket to receive the reply on.
- **dest_addr** – Destination IP address string.
- **ttl** – TTL value for this probe.
- **port** – UDP destination port for this probe.

Returns: A tuple `(addr, rtt_ms)` where *addr* is the responding router's IP address and *rtt_ms* is the round-trip time in milliseconds. Returns `(None, None)` if no reply is received within the timeout.

my_traceroute.**traceroute**(*destination: str*, *numeric: bool = False*, *nqueries: int = 3*, *summary: bool = False*) → None [source]

Run traceroute to *destination* and print results.

Parameters:
- **destination** – Hostname or IP address to trace the route to.
- **numeric** – If `True`, suppress reverse DNS lookups and print addresses numerically only (`-n` flag).
- **nqueries** – Number of UDP probes to send per TTL hop (`-q` flag).
- **summary** – If `True`, print a per-hop summary of unanswered probes (`-S` flag).