. TUGAS BESAR JARINGAN KOMPUTER: IMPLEMENTASI PROXY SERVER, SOCKET PROGRAMMING, DAN ANALISIS QOS MENGGUNAKAN WIRESHARK

Dosen Pengampu:

Dr. Viddi Mardiansyah, S.Si., M.T., CCNA., CCAI., CCST., MOS., SMIEEE.

Oleh:

Azzahra Indah	103012300238
Ahmad Refi Widi Katibin	103012300231

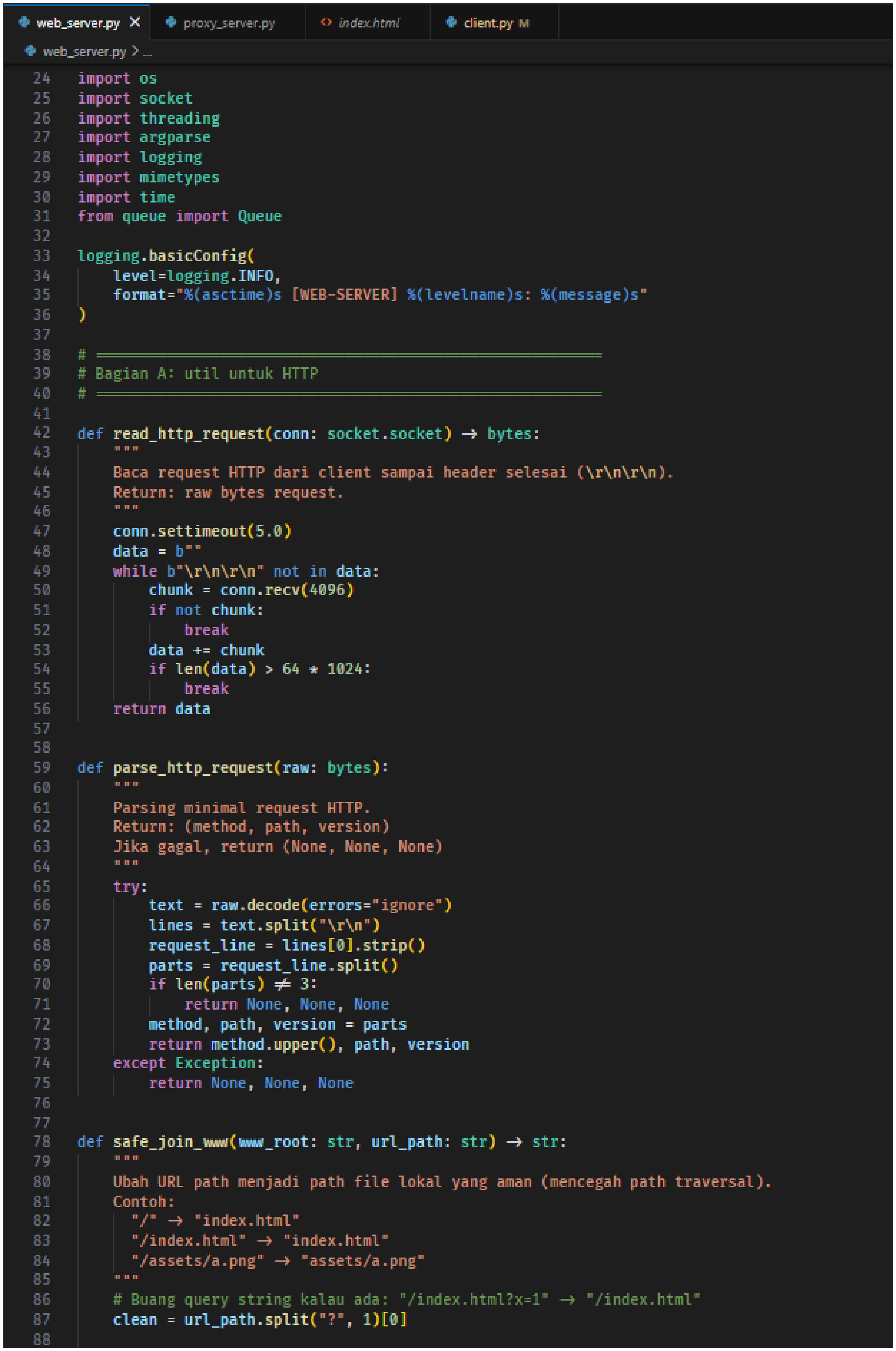
PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

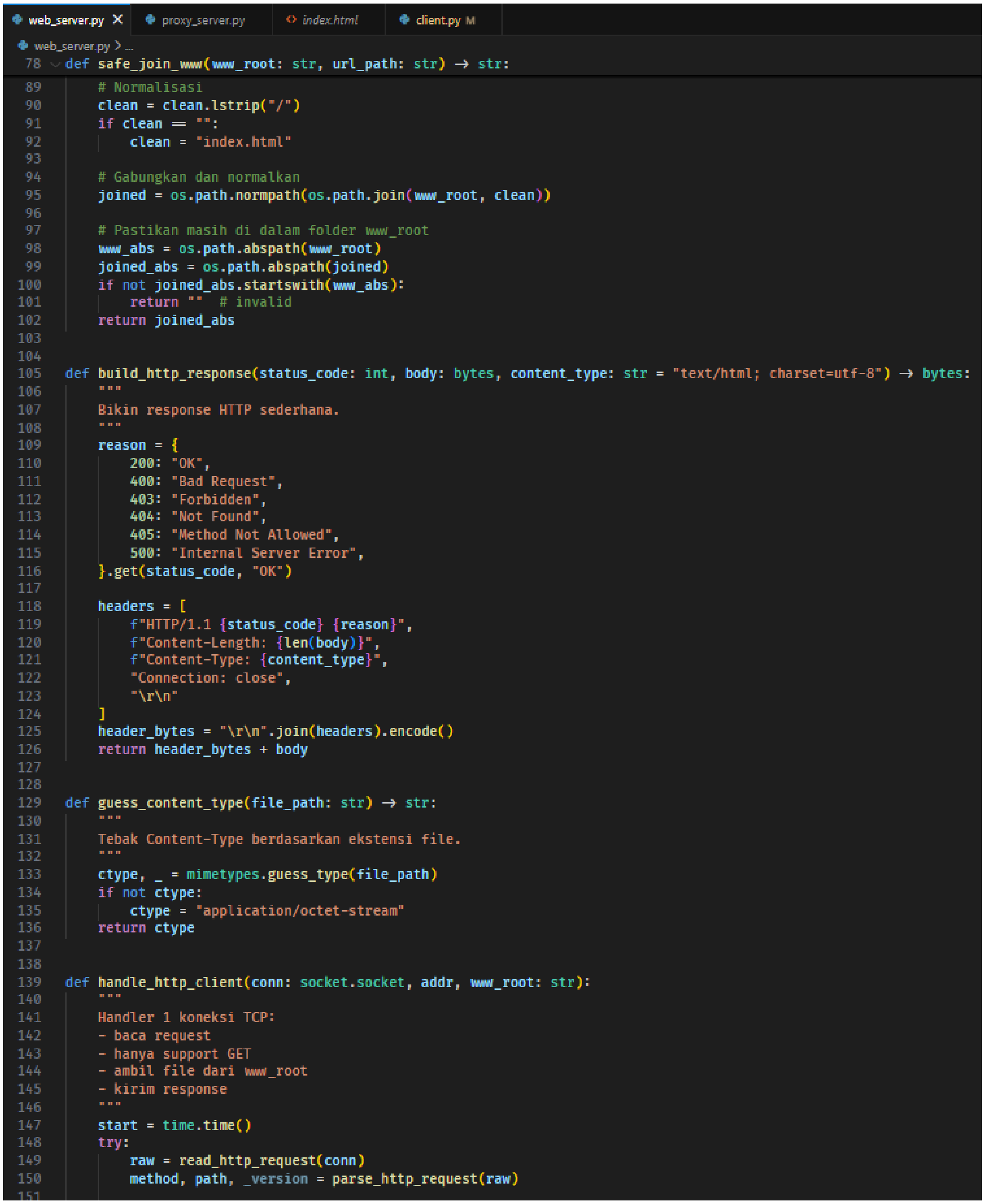
UNIVERSITAS TELKOM

2025

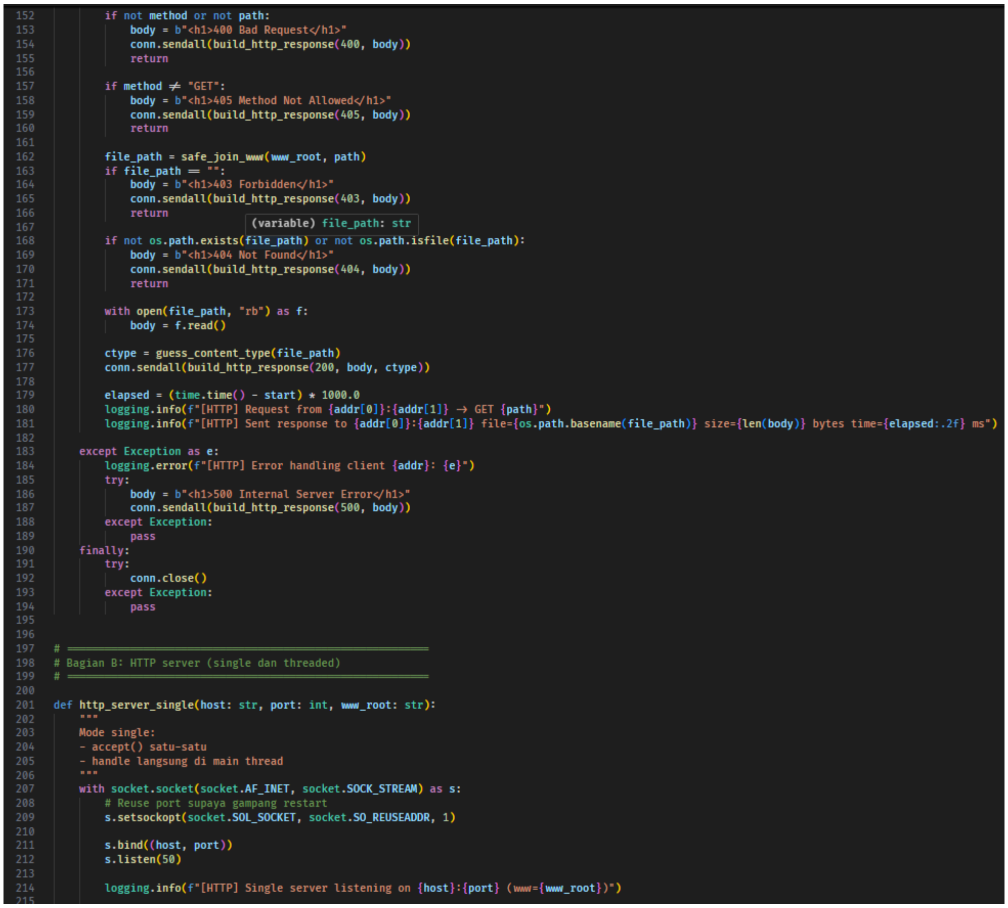
1. Pembagian Anggota dan File yang dikerjakan
    1.1. Azzahra Indah mengerjakan file web server dan proxy server
    1.2. Ahmad Refi Widi Katibin mengerjakan file client
2. Implementasi Socket Programming
    2.1. Memberikan code
        2.1.1. Web_server



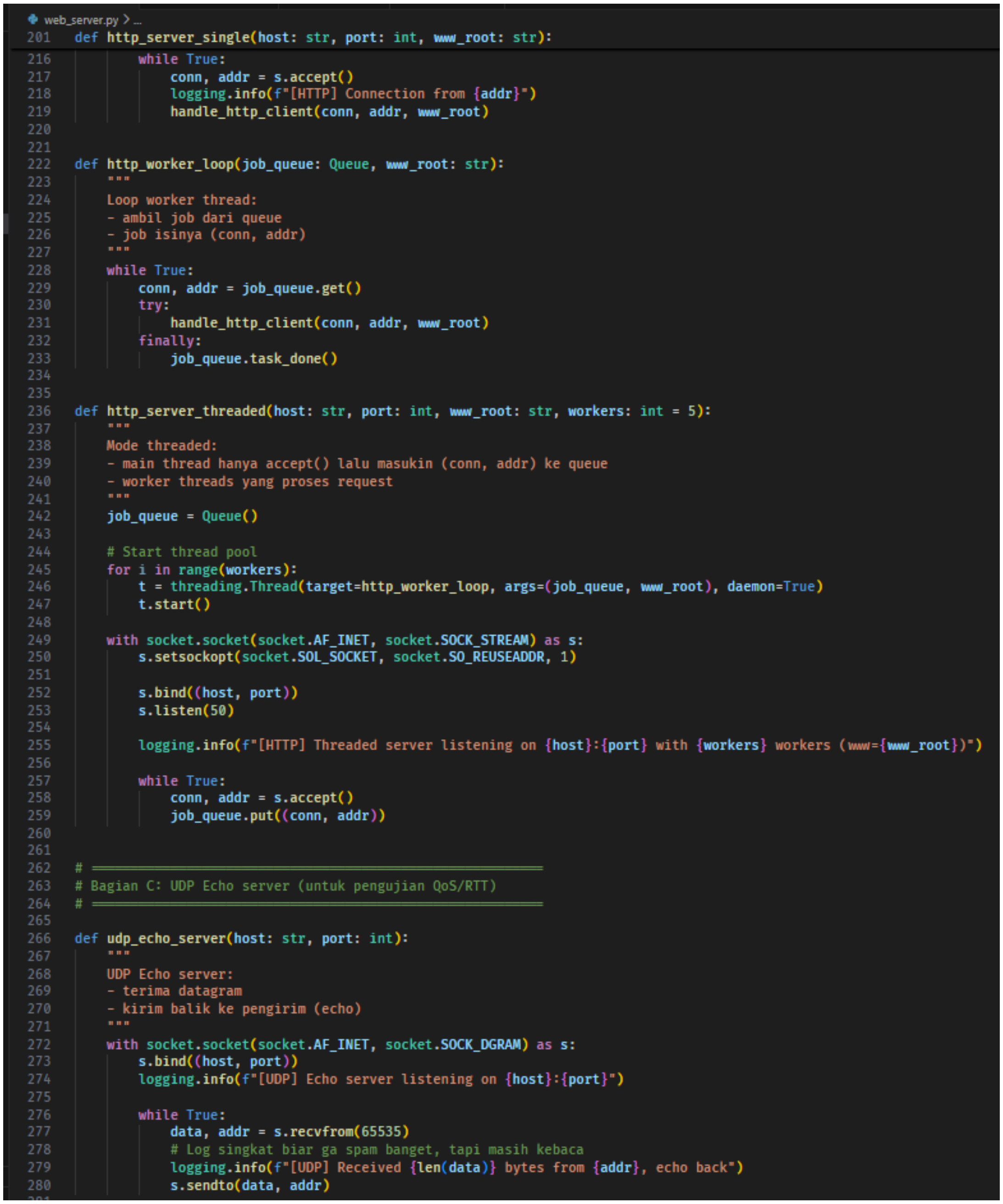
```python
import os
import socket
import threading
import argparse
import logging
import mimetypes
import time
from queue import Queue

logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s [WEB-SERVER] %(levelname)s: %(message)s"
)

# =========================================================
# Bagian A: util untuk HTTP
# =========================================================

def read_http_request(conn: socket.socket) -> bytes:
    """
    Baca request HTTP dari client sampai header selesai (\r\n\r\n).
    Return: raw bytes request.
    """
    conn.settimeout(5.0)
    data = b""
    while b"\r\n\r\n" not in data:
        chunk = conn.recv(4096)
        if not chunk:
            break
        data += chunk
        if len(data) > 64 * 1024:
            break
    return data


def parse_http_request(raw: bytes):
    """
    Parsing minimal request HTTP.
    Return: (method, path, version)
    Jika gagal, return (None, None, None)
    """
    try:
        text = raw.decode(errors="ignore")
        lines = text.split("\r\n")
        request_line = lines[0].strip()
        parts = request_line.split()
        if len(parts) != 3:
            return None, None, None
        method, path, version = parts
        return method.upper(), path, version
    except Exception:
        return None, None, None


def safe_join_www(www_root: str, url_path: str) -> str:
    """
    Ubah URL path menjadi path file lokal yang aman (mencegah path traversal).
    Contoh:
      "/" -> "index.html"
      "/index.html" -> "index.html"
      "/assets/a.png" -> "assets/a.png"
    """
    # Buang query string kalau ada: "/index.html?x=1" -> "/index.html"
    clean = url_path.split("?", 1)[0]

```



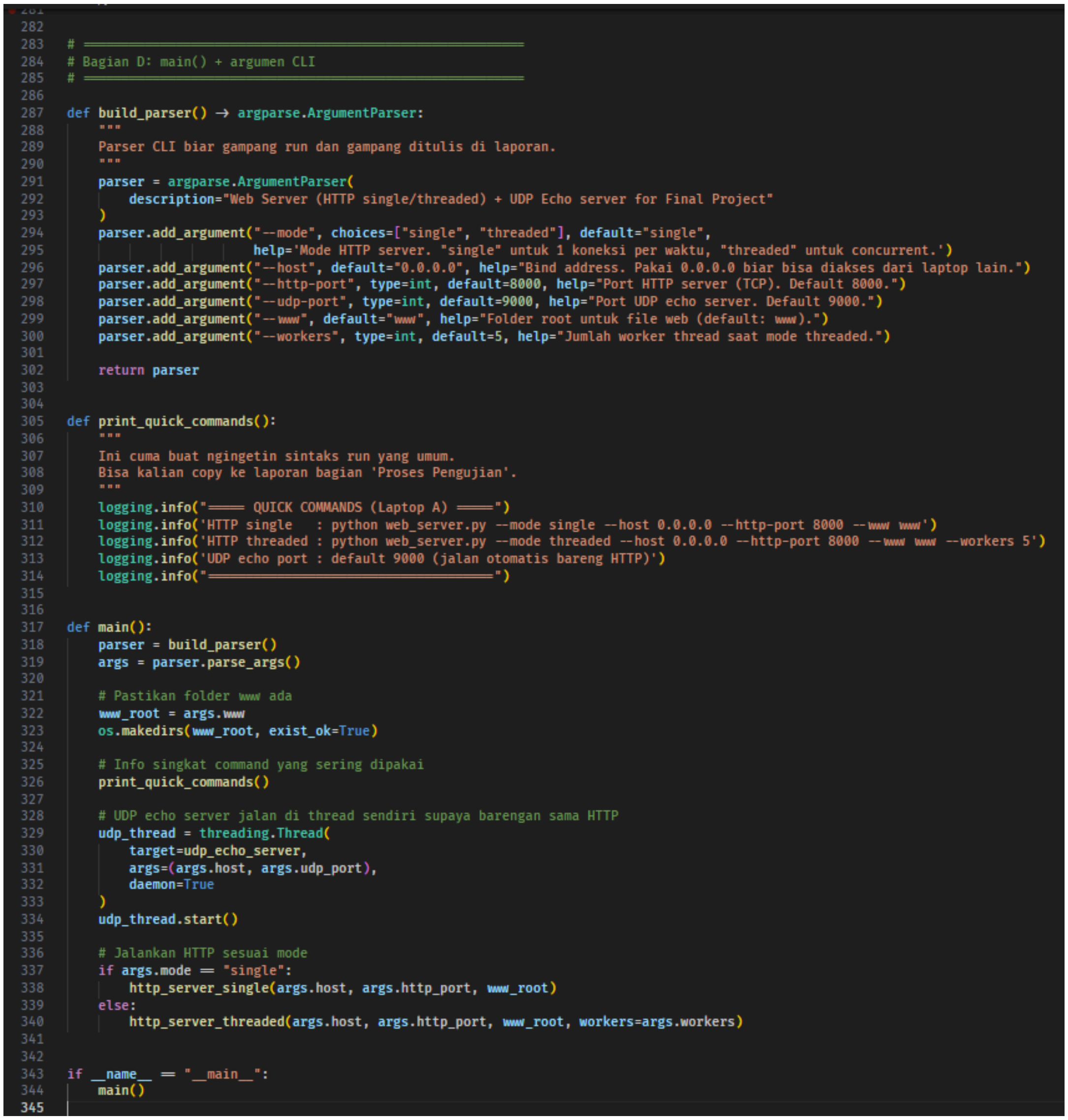
```python
def safe_join_www(www_root: str, url_path: str) -> str:
    # Normalisasi
    clean = clean.lstrip("/")
    if clean == "":
        clean = "index.html"

    # Gabungkan dan normalkan
    joined = os.path.normpath(os.path.join(www_root, clean))

    # Pastikan masih di dalam folder www_root
    www_abs = os.path.abspath(www_root)
    joined_abs = os.path.abspath(joined)
    if not joined_abs.startswith(www_abs):
        return ""  # invalid
    return joined_abs


def build_http_response(status_code: int, body: bytes, content_type: str = "text/html; charset=utf-8") -> bytes:
    """
    Bikin response HTTP sederhana.
    """
    reason = {
        200: "OK",
        400: "Bad Request",
        403: "Forbidden",
        404: "Not Found",
        405: "Method Not Allowed",
        500: "Internal Server Error",
    }.get(status_code, "OK")

    headers = [
        f"HTTP/1.1 {status_code} {reason}",
        f"Content-Length: {len(body)}",
        f"Content-Type: {content_type}",
        "Connection: close",
        "\r\n"
    ]
    header_bytes = "\r\n".join(headers).encode()
    return header_bytes + body


def guess_content_type(file_path: str) -> str:
    """
    Tebak Content-Type berdasarkan ekstensi file.
    """
    ctype, _ = mimetypes.guess_type(file_path)
    if not ctype:
        ctype = "application/octet-stream"
    return ctype


def handle_http_client(conn: socket.socket, addr, www_root: str):
    """
    Handler 1 koneksi TCP:
    - baca request
    - hanya support GET
    - ambil file dari www_root
    - kirim response
    """
    start = time.time()
    try:
        raw = read_http_request(conn)
        method, path, _version = parse_http_request(raw)

```



```python
        if not method or not path:
            body = b"<h1>400 Bad Request</h1>"
            conn.sendall(build_http_response(400, body))
            return

        if method ≠ "GET":
            body = b"<h1>405 Method Not Allowed</h1>"
            conn.sendall(build_http_response(405, body))
            return

        file_path = safe_join_www(www_root, path)
        if file_path == "":
            body = b"<h1>403 Forbidden</h1>"
            conn.sendall(build_http_response(403, body))
            return

        if not os.path.exists(file_path) or not os.path.isfile(file_path):
            body = b"<h1>404 Not Found</h1>"
            conn.sendall(build_http_response(404, body))
            return

        with open(file_path, "rb") as f:
            body = f.read()

        ctype = guess_content_type(file_path)
        conn.sendall(build_http_response(200, body, ctype))

        elapsed = (time.time() - start) * 1000.0
        logging.info(f"[HTTP] Request from {addr[0]}:{addr[1]} → GET {path}")
        logging.info(f"[HTTP] Sent response to {addr[0]}:{addr[1]} file={os.path.basename(file_path)} size={len(body)} bytes time={elapsed:.2f} ms")

    except Exception as e:
        logging.error(f"[HTTP] Error handling client {addr}: {e}")
        try:
            body = b"<h1>500 Internal Server Error</h1>"
            conn.sendall(build_http_response(500, body))
        except Exception:
            pass
    finally:
        try:
            conn.close()
        except Exception:
            pass


# ============================================================
# Bagian B: HTTP server (single dan threaded)
# ============================================================

def http_server_single(host: str, port: int, www_root: str):
    """
    Mode single:
    - accept() satu-satu
    - handle langsung di main thread
    """
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        # Reuse port supaya gampang restart
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

        s.bind((host, port))
        s.listen(50)

        logging.info(f"[HTTP] Single server listening on {host}:{port} (www={www_root})")
```

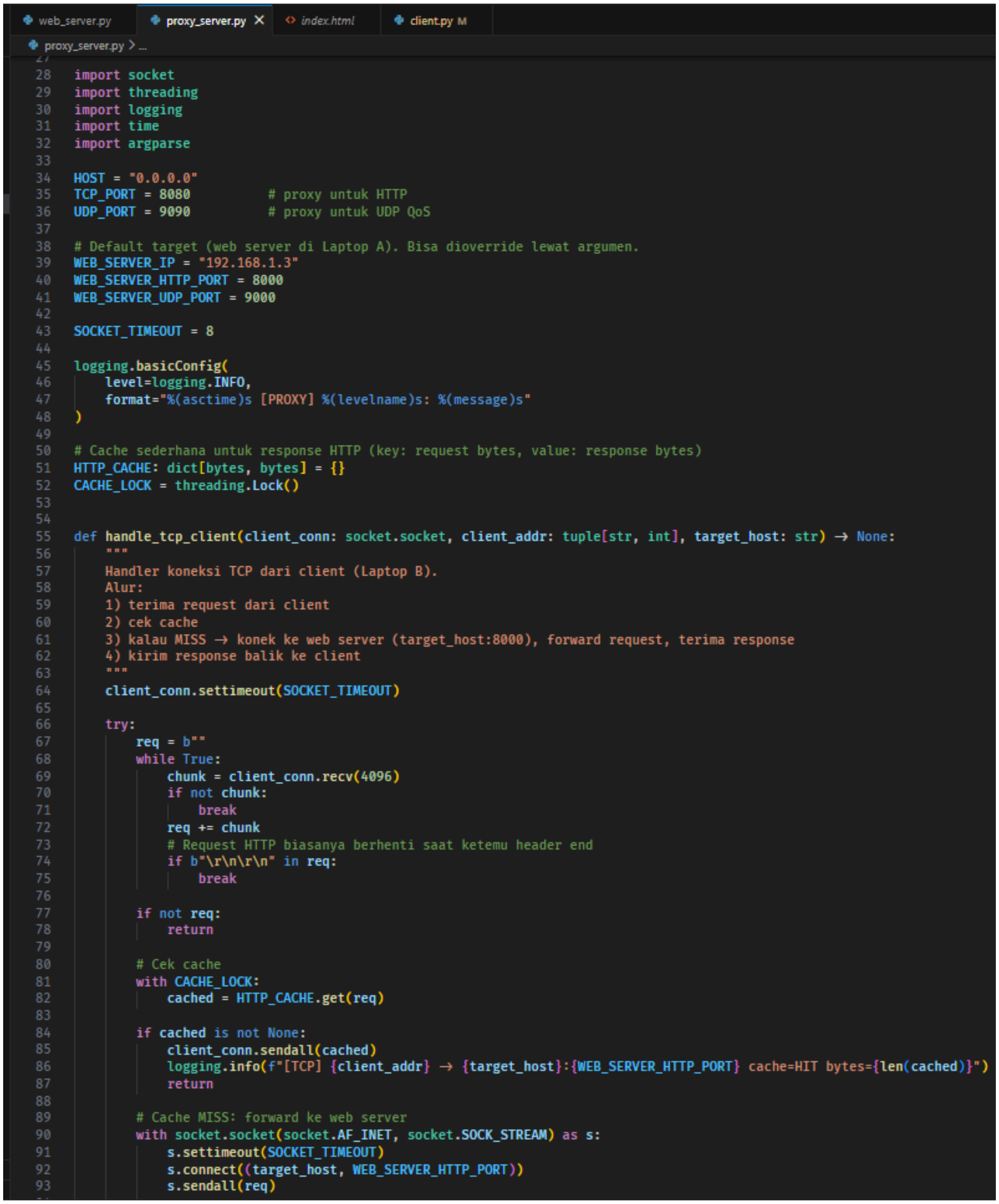


```python
def http_server_single(host: str, port: int, www_root: str):
        while True:
            conn, addr = s.accept()
            logging.info(f"[HTTP] Connection from {addr}")
            handle_http_client(conn, addr, www_root)


def http_worker_loop(job_queue: Queue, www_root: str):
    """
    Loop worker thread:
    - ambil job dari queue
    - job isinya (conn, addr)
    """
    while True:
        conn, addr = job_queue.get()
        try:
            handle_http_client(conn, addr, www_root)
        finally:
            job_queue.task_done()


def http_server_threaded(host: str, port: int, www_root: str, workers: int = 5):
    """
    Mode threaded:
    - main thread hanya accept() lalu masukin (conn, addr) ke queue
    - worker threads yang proses request
    """
    job_queue = Queue()

    # Start thread pool
    for i in range(workers):
        t = threading.Thread(target=http_worker_loop, args=(job_queue, www_root), daemon=True)
        t.start()

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

        s.bind((host, port))
        s.listen(50)

        logging.info(f"[HTTP] Threaded server listening on {host}:{port} with {workers} workers (www={www_root})")

        while True:
            conn, addr = s.accept()
            job_queue.put((conn, addr))


# ========================================================
# Bagian C: UDP Echo server (untuk pengujian QoS/RTT)
# ========================================================

def udp_echo_server(host: str, port: int):
    """
    UDP Echo server:
    - terima datagram
    - kirim balik ke pengirim (echo)
    """
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        s.bind((host, port))
        logging.info(f"[UDP] Echo server listening on {host}:{port}")

        while True:
            data, addr = s.recvfrom(65535)
            # Log singkat biar ga spam banget, tapi masih kebaca
            logging.info(f"[UDP] Received {len(data)} bytes from {addr}, echo back")
            s.sendto(data, addr)
```

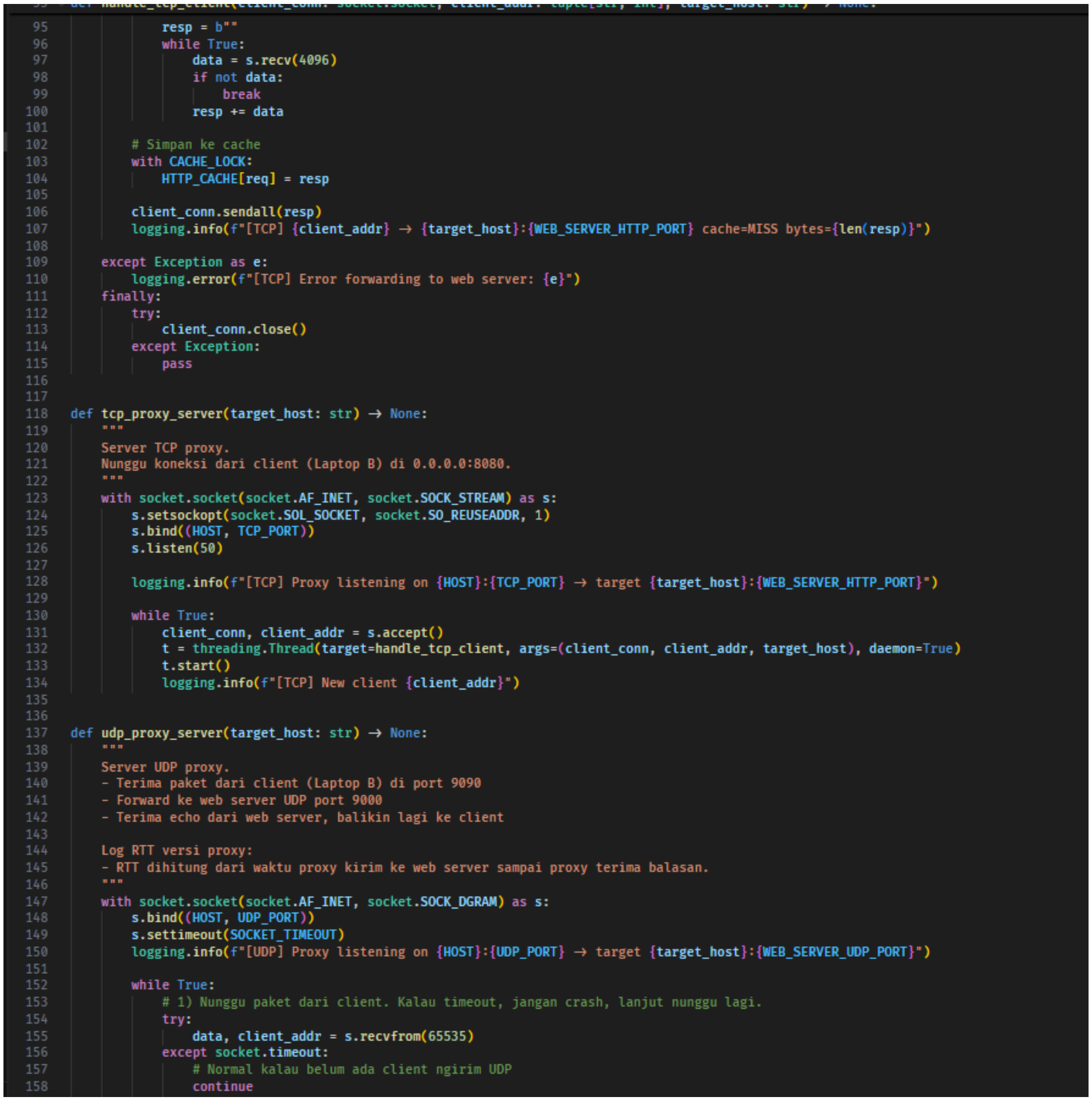


```python

# =========================================================
# Bagian D: main() + argumen CLI
# =========================================================

def build_parser() → argparse.ArgumentParser:
    """
    Parser CLI biar gampang run dan gampang ditulis di laporan.
    """
    parser = argparse.ArgumentParser(
        description="Web Server (HTTP single/threaded) + UDP Echo server for Final Project"
    )
    parser.add_argument("--mode", choices=["single", "threaded"], default="single",
                        help='Mode HTTP server. "single" untuk 1 koneksi per waktu, "threaded" untuk concurrent.')
    parser.add_argument("--host", default="0.0.0.0", help="Bind address. Pakai 0.0.0.0 biar bisa diakses dari laptop lain.")
    parser.add_argument("--http-port", type=int, default=8000, help="Port HTTP server (TCP). Default 8000.")
    parser.add_argument("--udp-port", type=int, default=9000, help="Port UDP echo server. Default 9000.")
    parser.add_argument("--www", default="www", help="Folder root untuk file web (default: www).")
    parser.add_argument("--workers", type=int, default=5, help="Jumlah worker thread saat mode threaded.")

    return parser


def print_quick_commands():
    """
    Ini cuma buat ngingetin sintaks run yang umum.
    Bisa kalian copy ke laporan bagian 'Proses Pengujian'.
    """
    logging.info("===== QUICK COMMANDS (Laptop A) =====")
    logging.info('HTTP single   : python web_server.py --mode single --host 0.0.0.0 --http-port 8000 --www www')
    logging.info('HTTP threaded : python web_server.py --mode threaded --host 0.0.0.0 --http-port 8000 --www www --workers 5')
    logging.info('UDP echo port : default 9000 (jalan otomatis bareng HTTP)')
    logging.info("=====================================")


def main():
    parser = build_parser()
    args = parser.parse_args()

    # Pastikan folder www ada
    www_root = args.www
    os.makedirs(www_root, exist_ok=True)

    # Info singkat command yang sering dipakai
    print_quick_commands()

    # UDP echo server jalan di thread sendiri supaya barengan sama HTTP
    udp_thread = threading.Thread(
        target=udp_echo_server,
        args=(args.host, args.udp_port),
        daemon=True
    )
    udp_thread.start()

    # Jalankan HTTP sesuai mode
    if args.mode == "single":
        http_server_single(args.host, args.http_port, www_root)
    else:
        http_server_threaded(args.host, args.http_port, www_root, workers=args.workers)


if __name__ == "__main__":
    main()

```

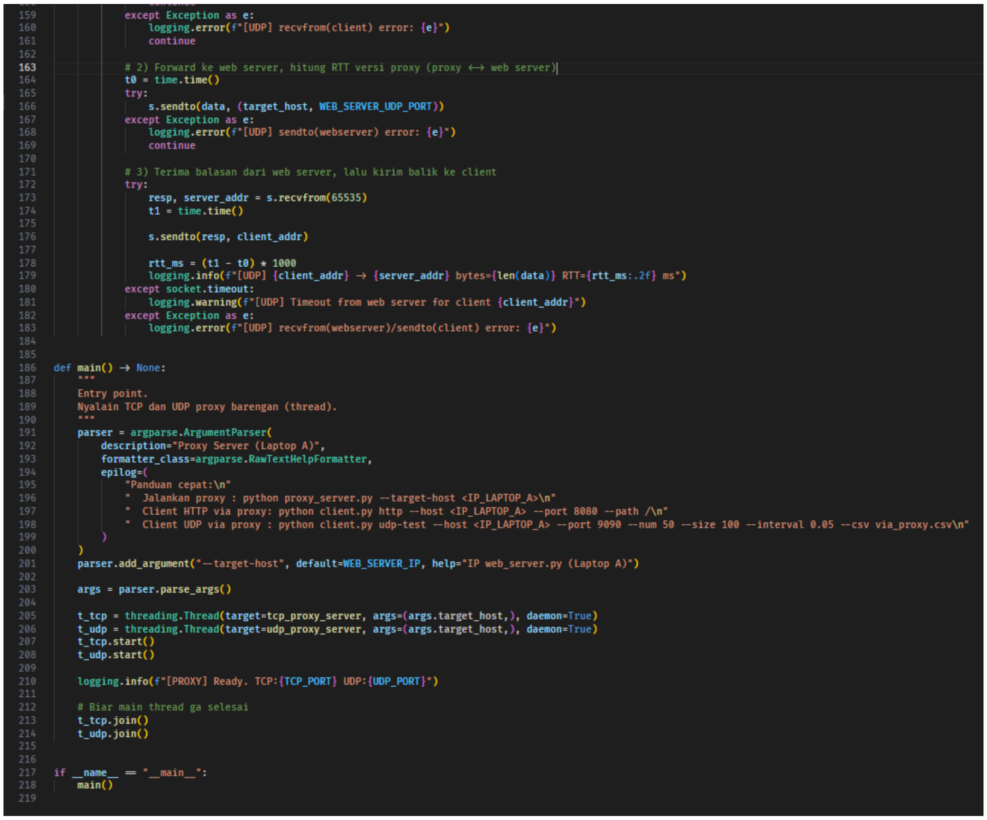
2.1.2. Proxy_server



```python
import socket
import threading
import logging
import time
import argparse

HOST = "0.0.0.0"
TCP_PORT = 8080          # proxy untuk HTTP
UDP_PORT = 9090          # proxy untuk UDP QoS

# Default target (web server di Laptop A). Bisa dioverride lewat argumen.
WEB_SERVER_IP = "192.168.1.3"
WEB_SERVER_HTTP_PORT = 8000
WEB_SERVER_UDP_PORT = 9000

SOCKET_TIMEOUT = 8

logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s [PROXY] %(levelname)s: %(message)s"
)

# Cache sederhana untuk response HTTP (key: request bytes, value: response bytes)
HTTP_CACHE: dict[bytes, bytes] = {}
CACHE_LOCK = threading.Lock()


def handle_tcp_client(client_conn: socket.socket, client_addr: tuple[str, int], target_host: str) → None:
    """
    Handler koneksi TCP dari client (Laptop B).
    Alur:
    1) terima request dari client
    2) cek cache
    3) kalau MISS → konek ke web server (target_host:8000), forward request, terima response
    4) kirim response balik ke client
    """
    client_conn.settimeout(SOCKET_TIMEOUT)

    try:
        req = b""
        while True:
            chunk = client_conn.recv(4096)
            if not chunk:
                break
            req += chunk
            # Request HTTP biasanya berhenti saat ketemu header end
            if b"\r\n\r\n" in req:
                break

        if not req:
            return

        # Cek cache
        with CACHE_LOCK:
            cached = HTTP_CACHE.get(req)

        if cached is not None:
            client_conn.sendall(cached)
            logging.info(f"[TCP] {client_addr} → {target_host}:{WEB_SERVER_HTTP_PORT} cache=HIT bytes={len(cached)}")
            return

        # Cache MISS: forward ke web server
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(SOCKET_TIMEOUT)
            s.connect((target_host, WEB_SERVER_HTTP_PORT))
            s.sendall(req)
```

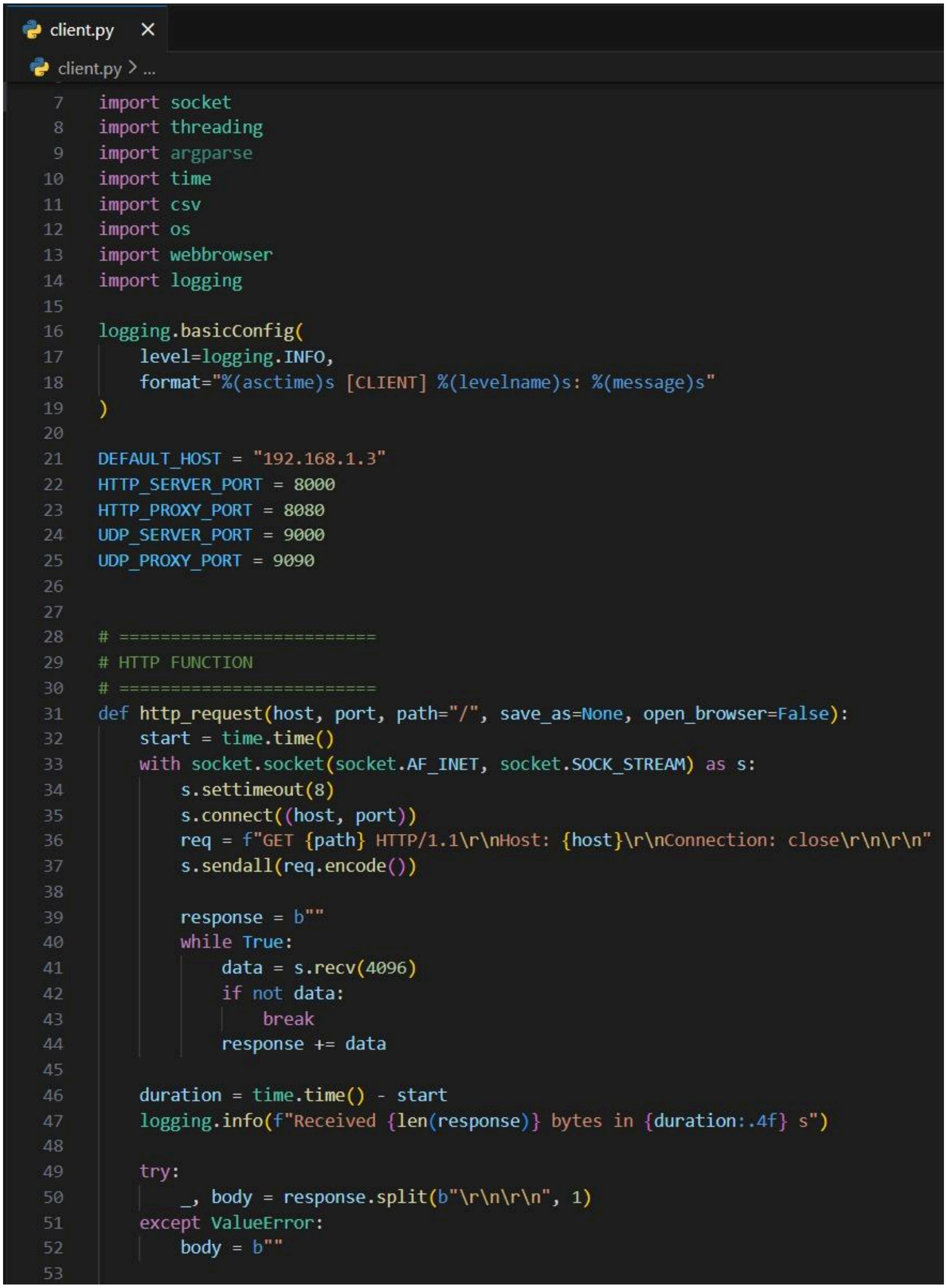


```python
def handle_tcp_client(client_conn: socket.socket, client_addr: tuple[str, int], target_host: str) → None:
            resp = b""
            while True:
                data = s.recv(4096)
                if not data:
                    break
                resp += data

        # Simpan ke cache
        with CACHE_LOCK:
            HTTP_CACHE[req] = resp

        client_conn.sendall(resp)
        logging.info(f"[TCP] {client_addr} → {target_host}:{WEB_SERVER_HTTP_PORT} cache=MISS bytes={len(resp)}")

    except Exception as e:
        logging.error(f"[TCP] Error forwarding to web server: {e}")
    finally:
        try:
            client_conn.close()
        except Exception:
            pass


def tcp_proxy_server(target_host: str) → None:
    """
    Server TCP proxy.
    Nunggu koneksi dari client (Laptop B) di 0.0.0.0:8080.
    """
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        s.bind((HOST, TCP_PORT))
        s.listen(50)

        logging.info(f"[TCP] Proxy listening on {HOST}:{TCP_PORT} → target {target_host}:{WEB_SERVER_HTTP_PORT}")

        while True:
            client_conn, client_addr = s.accept()
            t = threading.Thread(target=handle_tcp_client, args=(client_conn, client_addr, target_host), daemon=True)
            t.start()
            logging.info(f"[TCP] New client {client_addr}")


def udp_proxy_server(target_host: str) → None:
    """
    Server UDP proxy.
    - Terima paket dari client (Laptop B) di port 9090
    - Forward ke web server UDP port 9000
    - Terima echo dari web server, balikin lagi ke client

    Log RTT versi proxy:
    - RTT dihitung dari waktu proxy kirim ke web server sampai proxy terima balasan.
    """
    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        s.bind((HOST, UDP_PORT))
        s.settimeout(SOCKET_TIMEOUT)
        logging.info(f"[UDP] Proxy listening on {HOST}:{UDP_PORT} → target {target_host}:{WEB_SERVER_UDP_PORT}")

        while True:
            # 1) Nunggu paket dari client. Kalau timeout, jangan crash, lanjut nunggu lagi.
            try:
                data, client_addr = s.recvfrom(65535)
            except socket.timeout:
                # Normal kalau belum ada client ngirim UDP
                continue
```

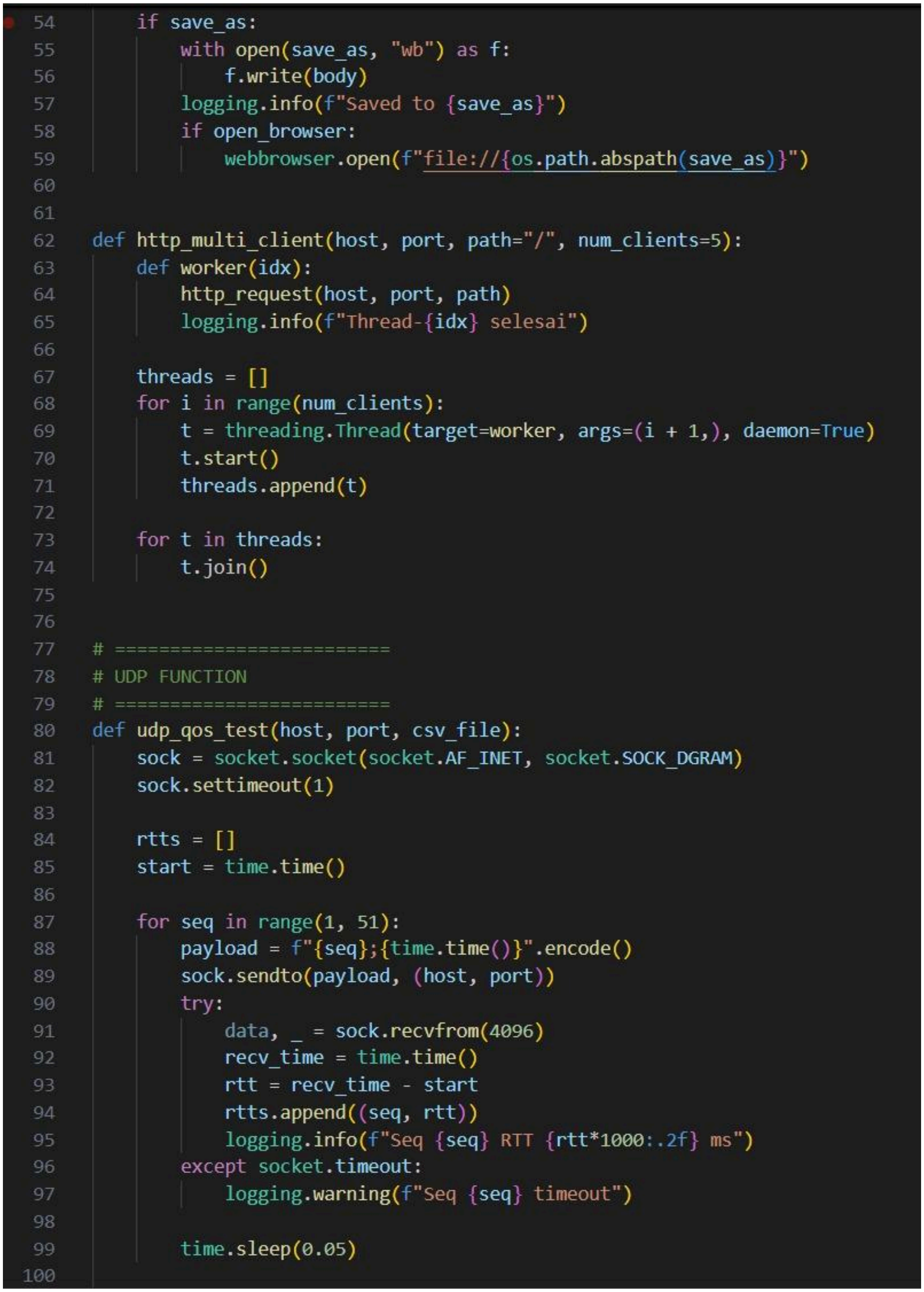


```python
            except Exception as e:
                logging.error(f"[UDP] recvfrom(client) error: {e}")
                continue

            # 2) Forward ke web server, hitung RTT versi proxy (proxy <-> web server)
            t0 = time.time()
            try:
                s.sendto(data, (target_host, WEB_SERVER_UDP_PORT))
            except Exception as e:
                logging.error(f"[UDP] sendto(webserver) error: {e}")
                continue

            # 3) Terima balasan dari web server, lalu kirim balik ke client
            try:
                resp, server_addr = s.recvfrom(65535)
                t1 = time.time()

                s.sendto(resp, client_addr)

                rtt_ms = (t1 - t0) * 1000
                logging.info(f"[UDP] {client_addr} -> {server_addr} bytes={len(data)} RTT={rtt_ms:.2f} ms")
            except socket.timeout:
                logging.warning(f"[UDP] Timeout from web server for client {client_addr}")
            except Exception as e:
                logging.error(f"[UDP] recvfrom(webserver)/sendto(client) error: {e}")


def main() -> None:
    """
    Entry point.
    Nyalain TCP dan UDP proxy barengan (thread).
    """
    parser = argparse.ArgumentParser(
        description="Proxy Server (Laptop A)",
        formatter_class=argparse.RawTextHelpFormatter,
        epilog=(
            "Panduan cepat:\n"
            "  Jalankan proxy : python proxy_server.py --target-host <IP_LAPTOP_A>\n"
            "  Client HTTP via proxy: python client.py http --host <IP_LAPTOP_A> --port 8080 --path /\n"
            "  Client UDP via proxy : python client.py udp-test --host <IP_LAPTOP_A> --port 9090 --num 50 --size 100 --interval 0.05 --csv via_proxy.csv\n"
        )
    )
    parser.add_argument("--target-host", default=WEB_SERVER_IP, help="IP web_server.py (Laptop A)")

    args = parser.parse_args()

    t_tcp = threading.Thread(target=tcp_proxy_server, args=(args.target_host,), daemon=True)
    t_udp = threading.Thread(target=udp_proxy_server, args=(args.target_host,), daemon=True)
    t_tcp.start()
    t_udp.start()

    logging.info(f"[PROXY] Ready. TCP:{TCP_PORT} UDP:{UDP_PORT}")

    # Biar main thread ga selesai
    t_tcp.join()
    t_udp.join()


if __name__ == "__main__":
    main()
```

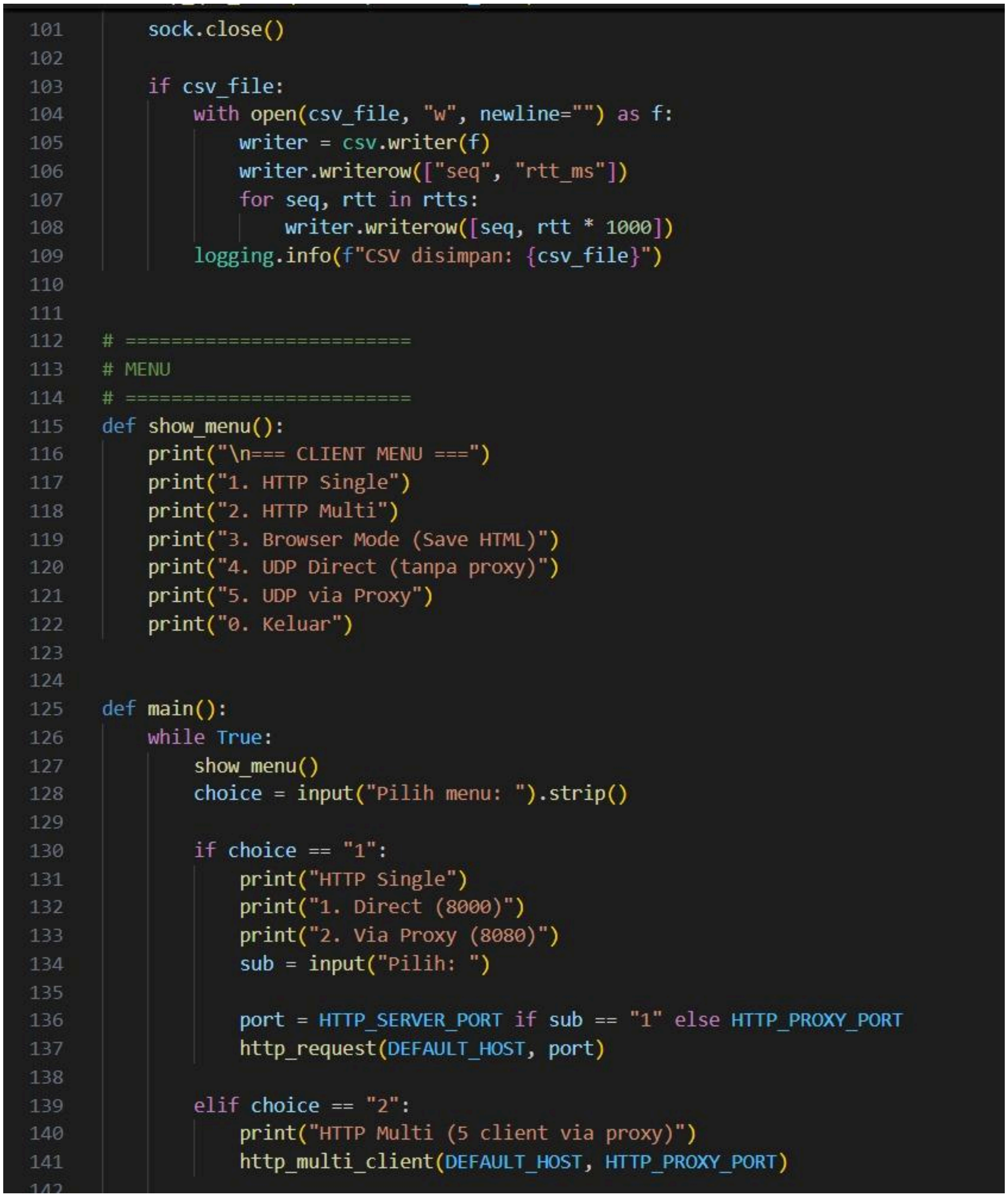
2.1.3. Client



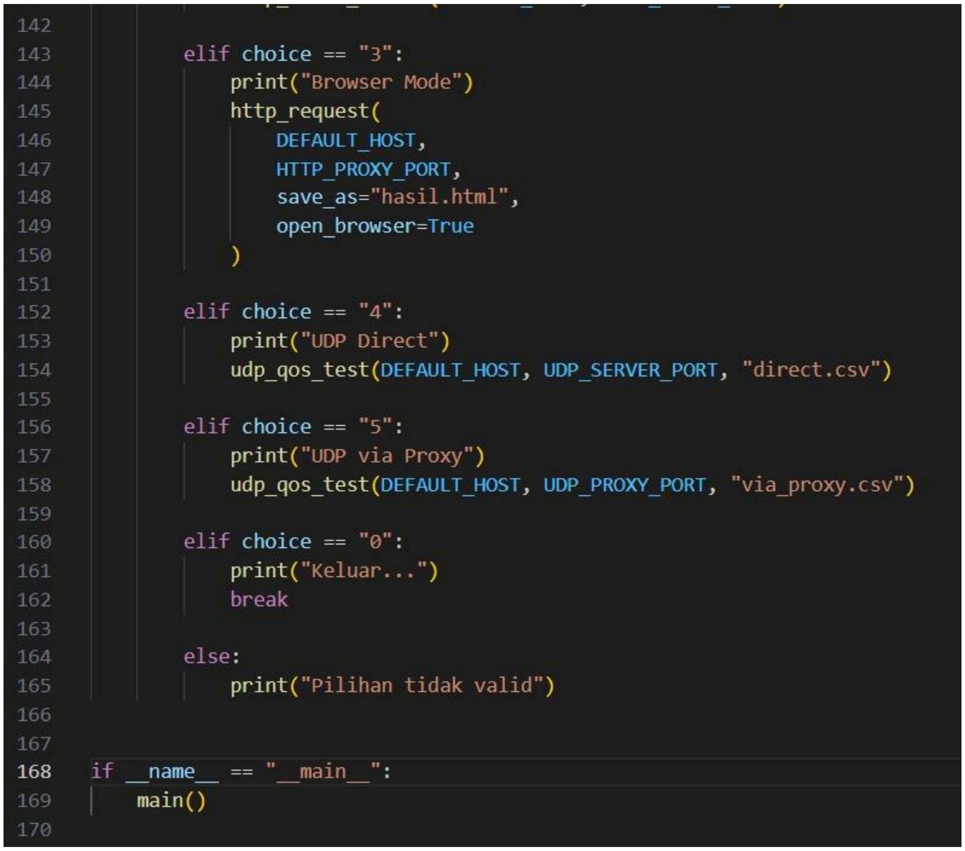
```python
import socket
import threading
import argparse
import time
import csv
import os
import webbrowser
import logging

logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s [CLIENT] %(levelname)s: %(message)s"
)

DEFAULT_HOST = "192.168.1.3"
HTTP_SERVER_PORT = 8000
HTTP_PROXY_PORT = 8080
UDP_SERVER_PORT = 9000
UDP_PROXY_PORT = 9090


# =========================
# HTTP FUNCTION
# =========================
def http_request(host, port, path="/", save_as=None, open_browser=False):
    start = time.time()
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.settimeout(8)
        s.connect((host, port))
        req = f"GET {path} HTTP/1.1\r\nHost: {host}\r\nConnection: close\r\n\r\n"
        s.sendall(req.encode())

        response = b""
        while True:
            data = s.recv(4096)
            if not data:
                break
            response += data

    duration = time.time() - start
    logging.info(f"Received {len(response)} bytes in {duration:.4f} s")

    try:
        _, body = response.split(b"\r\n\r\n", 1)
    except ValueError:
        body = b""

```



```python
    if save_as:
        with open(save_as, "wb") as f:
            f.write(body)
        logging.info(f"Saved to {save_as}")
        if open_browser:
            webbrowser.open(f"file://{os.path.abspath(save_as)}")


def http_multi_client(host, port, path="/", num_clients=5):
    def worker(idx):
        http_request(host, port, path)
        logging.info(f"Thread-{idx} selesai")

    threads = []
    for i in range(num_clients):
        t = threading.Thread(target=worker, args=(i + 1,), daemon=True)
        t.start()
        threads.append(t)

    for t in threads:
        t.join()


# ========================
# UDP FUNCTION
# ========================
def udp_qos_test(host, port, csv_file):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.settimeout(1)

    rtts = []
    start = time.time()

    for seq in range(1, 51):
        payload = f"{seq};{time.time()}".encode()
        sock.sendto(payload, (host, port))
        try:
            data, _ = sock.recvfrom(4096)
            recv_time = time.time()
            rtt = recv_time - start
            rtts.append((seq, rtt))
            logging.info(f"Seq {seq} RTT {rtt*1000:.2f} ms")
        except socket.timeout:
            logging.warning(f"Seq {seq} timeout")

        time.sleep(0.05)

```



```python
    sock.close()

    if csv_file:
        with open(csv_file, "w", newline="") as f:
            writer = csv.writer(f)
            writer.writerow(["seq", "rtt_ms"])
            for seq, rtt in rtts:
                writer.writerow([seq, rtt * 1000])
        logging.info(f"CSV disimpan: {csv_file}")


# ========================
# MENU
# ========================
def show_menu():
    print("\n=== CLIENT MENU ===")
    print("1. HTTP Single")
    print("2. HTTP Multi")
    print("3. Browser Mode (Save HTML)")
    print("4. UDP Direct (tanpa proxy)")
    print("5. UDP via Proxy")
    print("0. Keluar")


def main():
    while True:
        show_menu()
        choice = input("Pilih menu: ").strip()

        if choice == "1":
            print("HTTP Single")
            print("1. Direct (8000)")
            print("2. Via Proxy (8080)")
            sub = input("Pilih: ")

            port = HTTP_SERVER_PORT if sub == "1" else HTTP_PROXY_PORT
            http_request(DEFAULT_HOST, port)

        elif choice == "2":
            print("HTTP Multi (5 client via proxy)")
            http_multi_client(DEFAULT_HOST, HTTP_PROXY_PORT)
```



```
        elif choice == "3":
            print("Browser Mode")
            http_request(
                DEFAULT_HOST,
                HTTP_PROXY_PORT,
                save_as="hasil.html",
                open_browser=True
            )

        elif choice == "4":
            print("UDP Direct")
            udp_qos_test(DEFAULT_HOST, UDP_SERVER_PORT, "direct.csv")

        elif choice == "5":
            print("UDP via Proxy")
            udp_qos_test(DEFAULT_HOST, UDP_PROXY_PORT, "via_proxy.csv")

        elif choice == "0":
            print("Keluar...")
            break

        else:
            print("Pilihan tidak valid")


if __name__ == "__main__":
    main()
```

2.2. Penjelasan keterhubungan antara web server, proxy server, dan client

Pada sistem yang dibangun terdapat tiga komponen utama yaitu client, proxy server, dan web server yang saling terhubung dalam satu jaringan lokal. Client yang dijalankan pada Laptop B berperan sebagai pengirim permintaan, baik dalam bentuk permintaan HTTP maupun paket UDP untuk pengujian QoS. Web server yang berjalan di Laptop A berfungsi sebagai server utama yang menyediakan layanan HTTP pada port 8000 serta layanan UDP echo pada port 9000. Proxy server juga dijalankan di Laptop A dan berperan sebagai perantara antara client dan web server, di mana proxy menerima permintaan HTTP dari client pada port 8080 dan paket UDP pada port 9090, lalu meneruskannya ke web server. Selain meneruskan data, proxy server juga melakukan pencatatan aktivitas komunikasi serta menyediakan mekanisme cache sederhana untuk permintaan HTTP sehingga dapat diamati perbedaan cache HIT dan cache MISS. Dalam mode direct, client berkomunikasi langsung dengan web server, sedangkan dalam mode via proxy, seluruh permintaan client harus melewati proxy server terlebih dahulu sebelum diteruskan ke web server dan dikembalikan lagi ke client. Arsitektur ini memungkinkan dilakukan analisis perbandingan performa dan kualitas layanan jaringan antara komunikasi langsung dan komunikasi yang melalui proxy server.

3. Hasil dan Pembahasan

3.1. Hasil pengujian

3.1.1. Pengujian HTTP single

3.1.1.1. Tujuan Pengujian

Pengujian HTTP Single dilakukan untuk memastikan bahwa sistem client dan web server dapat berkomunikasi dengan baik menggunakan protokol HTTP dalam satu permintaan (single request). Selain itu, pengujian ini bertujuan untuk membandingkan proses komunikasi HTTP yang dilakukan secara langsung ke web server dengan komunikasi HTTP yang dilakukan melalui proxy server, sehingga peran proxy dalam alur komunikasi jaringan dapat diamati.

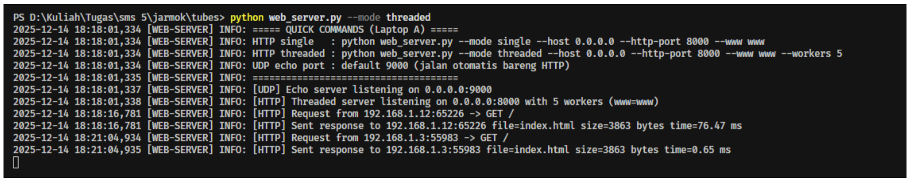
3.1.1.2. Proses Pengujian

Proses pengujian HTTP Single dilakukan dengan langkah-langkah sebagai berikut:

1. Web server dijalankan pada Laptop A dengan mode threaded dan mendengarkan koneksi HTTP pada port 8000.
2. Proxy server dijalankan pada Laptop A dan dikonfigurasi untuk meneruskan permintaan HTTP dari port 8080 ke web server pada port 8000.
3. Client dijalankan pada Laptop B melalui menu interaktif yang telah disediakan.
4. Client memilih menu HTTP Single, kemudian memilih salah satu dari dua skenario pengujian:
    a. Direct (port 8000): client mengirimkan satu permintaan HTTP GET langsung ke web server tanpa melalui proxy.
    b. Via Proxy (port 8080): client mengirimkan satu permintaan HTTP GET ke proxy server, kemudian proxy meneruskan permintaan tersebut ke web server.
5. Client menerima response dari server dan mencatat ukuran data serta waktu respon.
6. Aktivitas komunikasi HTTP diamati melalui output terminal pada web server, proxy server, dan client.

3.1.1.3. Hasil Pengamatan

3.1.1.3.1. Web server

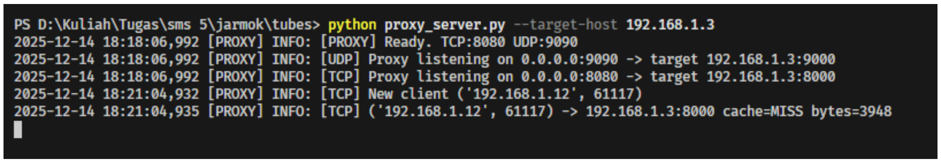


```
PS D:\Kuliah\Tugas\sms 5\jarmok\tubes> python web_server.py --mode threaded
2025-12-14 18:18:01,334 [WEB-SERVER] INFO: ===== QUICK COMMANDS (Laptop A) =====
2025-12-14 18:18:01,334 [WEB-SERVER] INFO: HTTP single   : python web_server.py --mode single --host 0.0.0.0 --http-port 8000 --www www
2025-12-14 18:18:01,334 [WEB-SERVER] INFO: HTTP threaded : python web_server.py --mode threaded --host 0.0.0.0 --http-port 8000 --www www --workers 5
2025-12-14 18:18:01,334 [WEB-SERVER] INFO: UDP echo port : default 9000 (jalan otomatis bareng HTTP)
2025-12-14 18:18:01,335 [WEB-SERVER] INFO: ====================================
2025-12-14 18:18:01,337 [WEB-SERVER] INFO: [UDP] Echo server listening on 0.0.0.0:9000
2025-12-14 18:18:01,338 [WEB-SERVER] INFO: [HTTP] Threaded server listening on 0.0.0.0:8000 with 5 workers (www=www)
2025-12-14 18:18:16,781 [WEB-SERVER] INFO: [HTTP] Request from 192.168.1.12:65226 -> GET /
2025-12-14 18:18:16,781 [WEB-SERVER] INFO: [HTTP] Sent response to 192.168.1.12:65226 file=index.html size=3863 bytes time=76.47 ms
2025-12-14 18:21:04,934 [WEB-SERVER] INFO: [HTTP] Request from 192.168.1.3:55983 -> GET /
2025-12-14 18:21:04,935 [WEB-SERVER] INFO: [HTTP] Sent response to 192.168.1.3:55983 file=index.html size=3863 bytes time=0.65 ms
```

Penjelasan:

- Web server aktif di port 8000
- Menerima request HTTP dari client (Laptop B)
- Memproses permintaan file index.html
- Mengirimkan response kembali ke client
- Log waktu menunjukkan lama proses request di sisi server

3.1.1.3.2. Proxy server

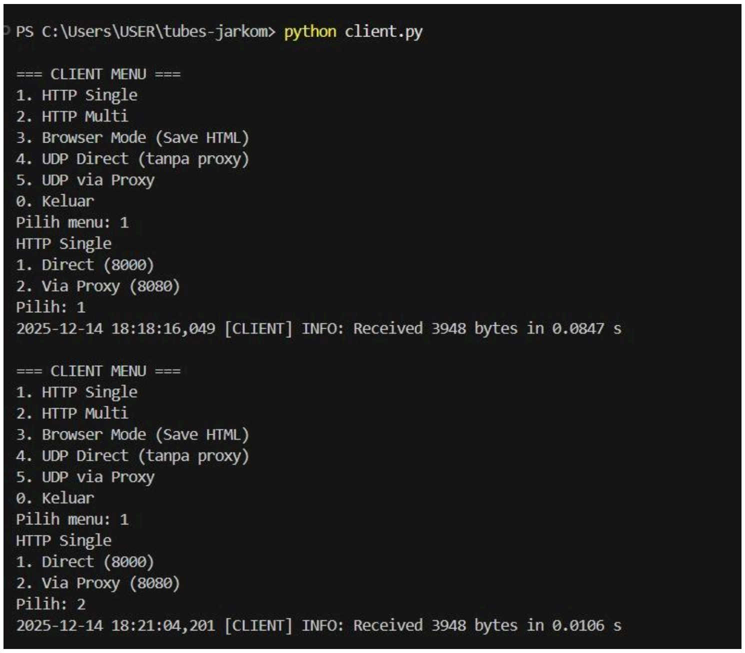


```
PS D:\Kuliah\Tugas\sms 5\jarmok\tubes> python proxy_server.py --target-host 192.168.1.3
2025-12-14 18:18:06,992 [PROXY] INFO: [PROXY] Ready. TCP:8080 UDP:9090
2025-12-14 18:18:06,992 [PROXY] INFO: [UDP] Proxy listening on 0.0.0.0:9090 -> target 192.168.1.3:9000
2025-12-14 18:18:06,992 [PROXY] INFO: [TCP] Proxy listening on 0.0.0.0:8080 -> target 192.168.1.3:8000
2025-12-14 18:21:04,932 [PROXY] INFO: [TCP] New client ('192.168.1.12', 61117)
2025-12-14 18:21:04,935 [PROXY] INFO: [TCP] ('192.168.1.12', 61117) -> 192.168.1.3:8000 cache=MISS bytes=3948
```

Penjelasan:

- Proxy aktif di port 8080
- Menerima request dari client
- Meneruskan request ke web server di port 8000
- Status cache=MISS berarti response belum ada di cache
- Jika request yang sama dikirim ulang, status bisa berubah menjadi cache=HIT

3.1.1.3.3. Client



```
PS C:\Users\USER\tubes-jarkom> python client.py

=== CLIENT MENU ===
1. HTTP Single
2. HTTP Multi
3. Browser Mode (Save HTML)
4. UDP Direct (tanpa proxy)
5. UDP via Proxy
0. Keluar
Pilih menu: 1
HTTP Single
1. Direct (8000)
2. Via Proxy (8080)
Pilih: 1
2025-12-14 18:18:16,049 [CLIENT] INFO: Received 3948 bytes in 0.0847 s

=== CLIENT MENU ===
1. HTTP Single
2. HTTP Multi
3. Browser Mode (Save HTML)
4. UDP Direct (tanpa proxy)
5. UDP via Proxy
0. Keluar
Pilih menu: 1
HTTP Single
1. Direct (8000)
2. Via Proxy (8080)
Pilih: 2
2025-12-14 18:21:04,201 [CLIENT] INFO: Received 3948 bytes in 0.0106 s
```

Penjelasan:

- Client mengirim request ke proxy server
- Proxy meneruskan request ke web server
- Response dikembalikan ke client
- Waktu respon bisa lebih cepat jika cache proxy aktif

3.1.1.3.4. Kesimpulan

Berdasarkan hasil pengujian yang dilakukan, client berhasil menerima response HTTP dari web server baik pada skenario direct maupun melalui proxy. Web server menampilkan log penerimaan permintaan HTTP GET dan pengiriman response ke client. Pada skenario melalui proxy, proxy server juga menampilkan log penerimaan client serta status cache saat meneruskan permintaan ke web server.

Ukuran response yang diterima pada kedua skenario menunjukkan nilai yang sama, yang menandakan bahwa data yang dikirimkan oleh web server tidak mengalami perubahan. Perbedaan yang diamati terletak pada jalur komunikasi jaringan, di mana pada skenario melalui proxy terdapat tambahan proses penerusan request serta mekanisme cache pada proxy server.

3.1.2. Pengujian HTTP Multi

3.1.2.1. Tujuan Pengujian

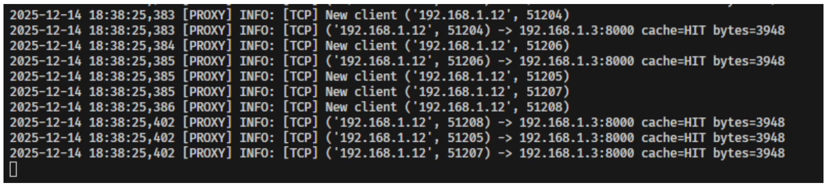
Pengujian HTTP Multi bertujuan untuk mengetahui kemampuan sistem dalam menangani beberapa permintaan HTTP secara bersamaan. Pengujian ini dilakukan untuk melihat bagaimana client dengan mekanisme multithreading dapat mengirimkan banyak request secara paralel, serta bagaimana proxy server dan web server merespons kondisi tersebut. Selain itu, pengujian ini juga bertujuan untuk mengamati peran cache pada proxy server dalam mengurangi beban kerja web server.

3.1.2.2. Proses Pengujian

Pengujian dilakukan dengan langkah-langkah sebagai berikut. Web server dijalankan terlebih dahulu pada Laptop A menggunakan mode threaded agar mampu melayani beberapa koneksi secara bersamaan. Setelah itu, proxy server dijalankan pada Laptop A dengan konfigurasi TCP port 8080 dan target menuju web server pada port 8000. Selanjutnya, pada Laptop B client dijalankan dan menu HTTP Multi dipilih. Client kemudian mengirimkan lima request HTTP secara paralel melalui proxy server menuju web server. Seluruh proses dijalankan setelah sistem dalam kondisi aktif dan siap menerima koneksi

3.1.2.3. Hasil Pengamatan

3.1.2.3.1. Proxy server



```
2025-12-14 18:38:25,383 [PROXY] INFO: [TCP] New client ('192.168.1.12', 51204)
2025-12-14 18:38:25,383 [PROXY] INFO: [TCP] ('192.168.1.12', 51204) -> 192.168.1.3:8000 cache=HIT bytes=3948
2025-12-14 18:38:25,384 [PROXY] INFO: [TCP] New client ('192.168.1.12', 51206)
2025-12-14 18:38:25,385 [PROXY] INFO: [TCP] ('192.168.1.12', 51206) -> 192.168.1.3:8000 cache=HIT bytes=3948
2025-12-14 18:38:25,385 [PROXY] INFO: [TCP] New client ('192.168.1.12', 51205)
2025-12-14 18:38:25,385 [PROXY] INFO: [TCP] New client ('192.168.1.12', 51207)
2025-12-14 18:38:25,386 [PROXY] INFO: [TCP] New client ('192.168.1.12', 51208)
2025-12-14 18:38:25,402 [PROXY] INFO: [TCP] ('192.168.1.12', 51208) -> 192.168.1.3:8000 cache=HIT bytes=3948
2025-12-14 18:38:25,402 [PROXY] INFO: [TCP] ('192.168.1.12', 51205) -> 192.168.1.3:8000 cache=HIT bytes=3948
2025-12-14 18:38:25,402 [PROXY] INFO: [TCP] ('192.168.1.12', 51207) -> 192.168.1.3:8000 cache=HIT bytes=3948
```

Penjelasan:

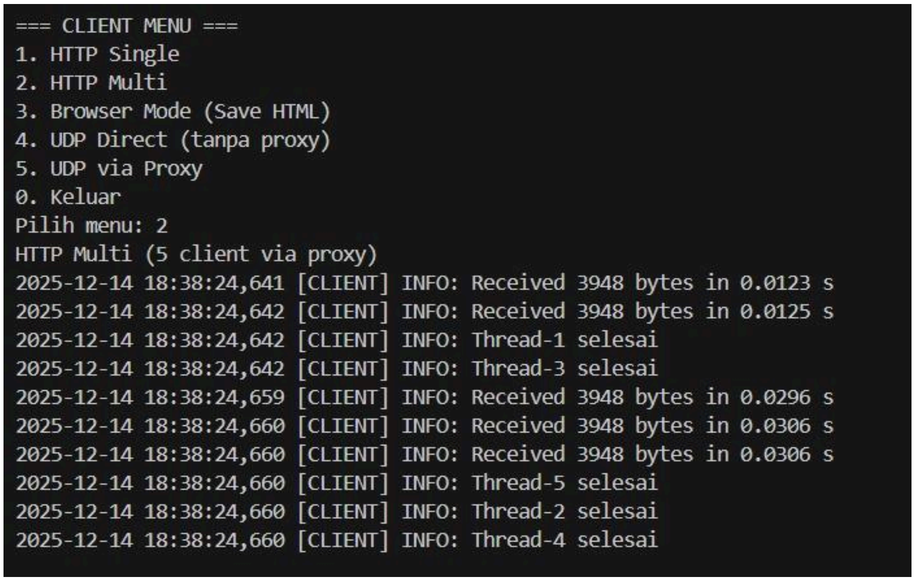
- Proxy menerima beberapa koneksi TCP hampir bersamaan dari client.
- Semua request dilayani dari cache (HIT), bukan diteruskan ulang ke web server.
- Ukuran response konsisten (3948 bytes), sesuai file HTML yang sama.

- Ini membuktikan fungsi cache proxy berjalan dengan benar, dan HTTP multi memanfaatkan cache.

3.1.2.3.2. Web server

Pada HTTP Multi, request melewati proxy. Karena response sudah pernah diminta sebelumnya, proxy mengambil dari cache. Akibatnya Web server tidak diakses ulang, maka tidak muncul log baru di terminal web server.

3.1.2.3.3. Client



```
=== CLIENT MENU ===
1. HTTP Single
2. HTTP Multi
3. Browser Mode (Save HTML)
4. UDP Direct (tanpa proxy)
5. UDP via Proxy
0. Keluar
Pilih menu: 2
HTTP Multi (5 client via proxy)
2025-12-14 18:38:24,641 [CLIENT] INFO: Received 3948 bytes in 0.0123 s
2025-12-14 18:38:24,642 [CLIENT] INFO: Received 3948 bytes in 0.0125 s
2025-12-14 18:38:24,642 [CLIENT] INFO: Thread-1 selesai
2025-12-14 18:38:24,642 [CLIENT] INFO: Thread-3 selesai
2025-12-14 18:38:24,659 [CLIENT] INFO: Received 3948 bytes in 0.0296 s
2025-12-14 18:38:24,660 [CLIENT] INFO: Received 3948 bytes in 0.0306 s
2025-12-14 18:38:24,660 [CLIENT] INFO: Received 3948 bytes in 0.0306 s
2025-12-14 18:38:24,660 [CLIENT] INFO: Thread-5 selesai
2025-12-14 18:38:24,660 [CLIENT] INFO: Thread-2 selesai
2025-12-14 18:38:24,660 [CLIENT] INFO: Thread-4 selesai
```

Penjelasan:

- Client berhasil menjalankan 5 request HTTP secara paralel menggunakan thread.
- Setiap thread mengirim request GET ke proxy (port 8080).
- Semua thread menerima response HTML dengan ukuran yang sama yaitu 3948 bytes, menandakan file yang sama berhasil dikirim.
- Waktu respon berbeda tipis antar thread, wajar karena eksekusi paralel.
- Log Thread-x selesai menandakan setiap thread sudah selesai menjalankan request-nya.

3.1.2.3.4. Kesimpulan

Berdasarkan hasil pengujian, client berhasil mengirimkan lima request HTTP secara bersamaan dan seluruh request mendapatkan response dengan ukuran file yang sama.

Waktu respon antar request relatif berdekatan, menandakan bahwa mekanisme multithreading pada client berjalan dengan baik. Pada sisi proxy server, terlihat bahwa seluruh request dilayani dengan status cache HIT, yang berarti proxy tidak meneruskan request ke web server melainkan langsung mengirimkan response dari cache. Akibatnya, pada sisi web server tidak terlihat adanya request baru selama pengujian HTTP Multi berlangsung. Hal ini menunjukkan bahwa cache pada proxy server berfungsi dengan baik dan mampu mengurangi beban web server saat menerima banyak request secara paralel.

3.1.3. Pengujian Browser Mode

3.1.3.1. Tujuan Pengujian

Pengujian browser mode bertujuan untuk memastikan bahwa client mampu menerima response HTTP dari web server melalui proxy, kemudian menyimpan body HTML ke dalam sebuah file lokal dan menampilkannya menggunakan browser. Pengujian ini juga membuktikan bahwa alur komunikasi HTTP client → proxy server → web server berjalan dengan benar hingga konten dapat dirender oleh browser.

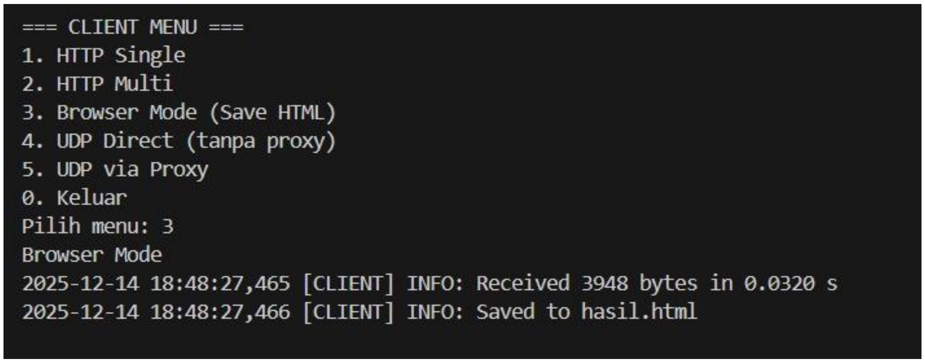
3.1.3.2. Proses Pengujian

Proses pengujian Browser Mode dilakukan dengan langkah-langkah sebagai berikut:

1. Web server dijalankan pada Laptop A dalam mode threaded dan mendengarkan koneksi HTTP pada port 8000.
2. Proxy server dijalankan pada Laptop A dengan TCP proxy di port 8080 dan diarahkan ke web server port 8000.
3. Client dijalankan pada Laptop B dan memilih menu Browser Mode (Save HTML).
4. Client mengirim request HTTP GET ke proxy server (port 8080).
5. Proxy meneruskan request tersebut ke web server, menerima response HTML, lalu mengirimkannya kembali ke client.
6. Client menyimpan body HTML ke file hasil.html dan otomatis membuka file tersebut menggunakan browser.

3.1.3.3. Hasil Pengamatan

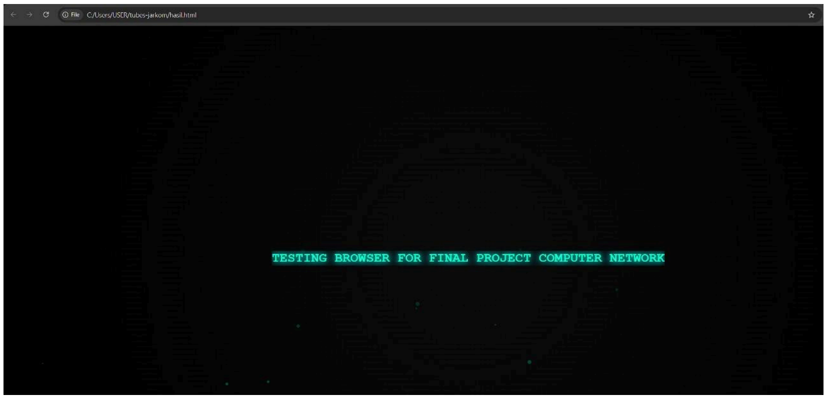
3.1.3.3.1. Client



```
=== CLIENT MENU ===
1. HTTP Single
2. HTTP Multi
3. Browser Mode (Save HTML)
4. UDP Direct (tanpa proxy)
5. UDP via Proxy
0. Keluar
Pilih menu: 3
Browser Mode
2025-12-14 18:48:27,465 [CLIENT] INFO: Received 3948 bytes in 0.0320 s
2025-12-14 18:48:27,466 [CLIENT] INFO: Saved to hasil.html
```

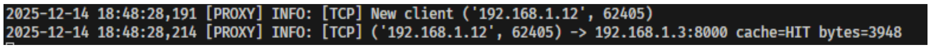
Terminal client menampilkan informasi bahwa response HTTP berhasil diterima dengan ukuran 3948 bytes dan file berhasil disimpan ke hasil.html. Hal ini menandakan proses request, penerimaan response, dan penyimpanan file berjalan tanpa error.

3.1.3.3.1.1. Browser menampilkan halaman HTML


File
C:/Users/USER/tubes-jarkom/hasil.html
TESTING BROWSER FOR FINAL PROJECT COMPUTER NETWORK

Browser berhasil membuka file hasil.html yang berisi konten HTML hasil response dari web server.

3.1.3.3.2. Proxy server



```
2025-12-14 18:48:28,191 [PROXY] INFO: [TCP] New client ('192.168.1.12', 62405)
2025-12-14 18:48:28,214 [PROXY] INFO: [TCP] ('192.168.1.12', 62405) -> 192.168.1.3:8000 cache=HIT bytes=3948
```

Terminal proxy menunjukkan adanya koneksi TCP baru dari client ke proxy, kemudian diteruskan ke web server dengan status cache HIT. Ini berarti proxy tidak perlu mengambil ulang data dari web server karena response sudah tersedia di cache, sehingga proses menjadi lebih cepat.

3.1.3.3.3. Kesimpulan

Pengujian browser mode menunjukkan bahwa client berhasil menerima response HTTP dari web server melalui proxy server, menyimpan body HTML ke dalam file lokal, dan menampilkannya dengan benar menggunakan browser. Mekanisme ini membuktikan bahwa alur komunikasi HTTP berjalan secara end-to-end mulai dari client, diteruskan oleh proxy, hingga diproses oleh web server. Selain itu, penggunaan cache pada proxy memungkinkan response dikirim tanpa selalu mengakses web server, sehingga meningkatkan efisiensi dan mempercepat waktu respon. Secara keseluruhan, browser mode berfungsi sesuai dengan tujuan pengujian dan mendukung validasi integrasi antara client, proxy server, dan web server.

3.1.4. Pengujian UDP Direct (Tanpa Proxy)

3.1.4.1. Tujuan Pengujian

Pengujian UDP direct dilakukan untuk mengukur kualitas komunikasi UDP antara client dan web server tanpa melalui proxy. Parameter yang diamati meliputi Round Trip Time (RTT), packet loss, jitter, dan throughput sebagai acuan performa jaringan dasar sebelum dibandingkan dengan pengujian UDP melalui proxy.

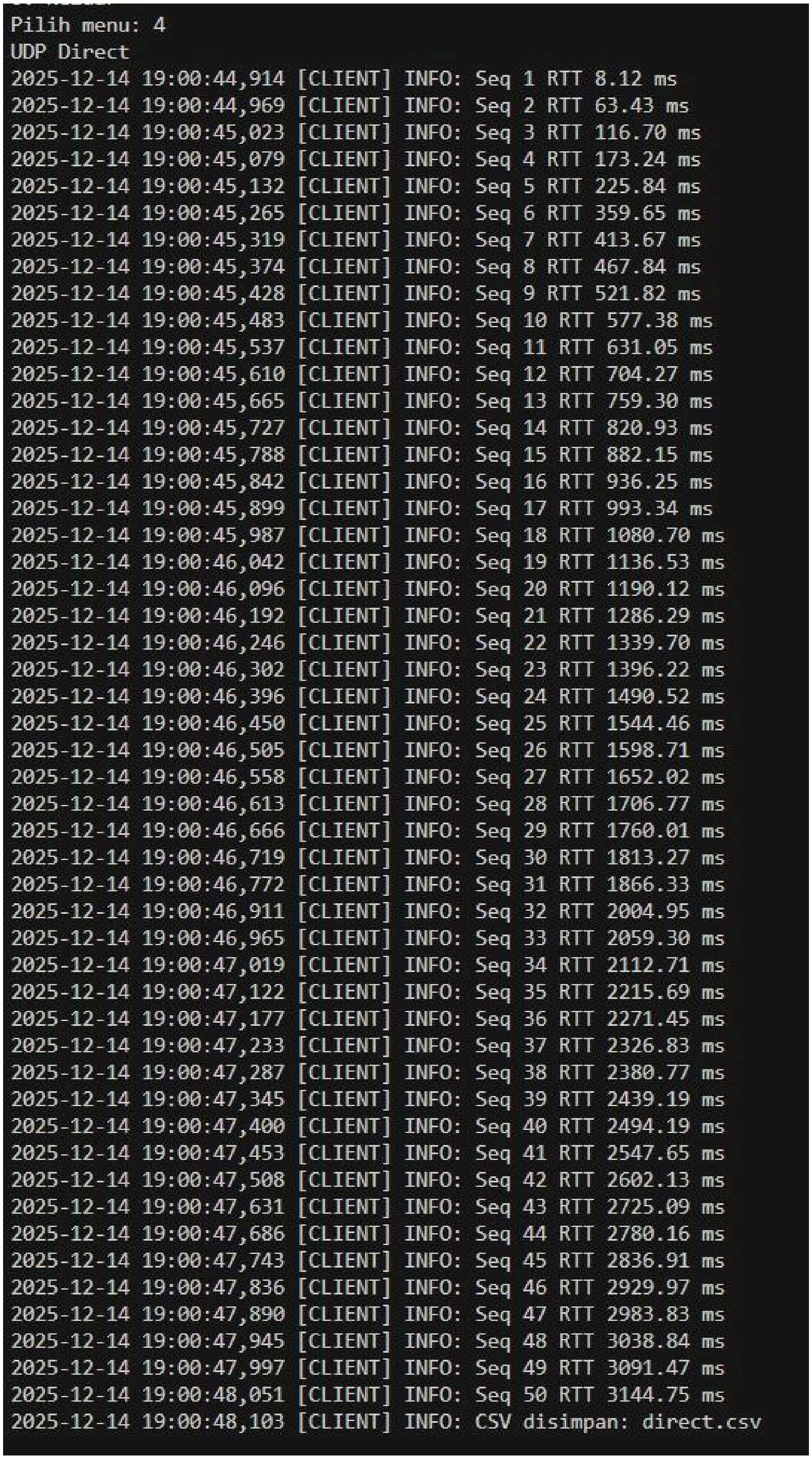
3.1.4.2. Proses Pengujian

Proses pengujian UDP Direct dilakukan dengan langkah-langkah sebagai berikut:

1. Web server dijalankan pada Laptop A dengan UDP echo server aktif di port 9000.
2. Client dijalankan pada Laptop B menggunakan menu UDP Direct (tanpa proxy).
3. Client mengirimkan sejumlah paket UDP secara berurutan ke web server dengan ukuran paket dan interval tertentu.
4. Web server menerima setiap paket dan langsung mengirimkan kembali paket tersebut ke client sebagai echo.
5. Client mencatat waktu kirim dan waktu terima untuk menghitung RTT, kemudian menyimpan hasil pengujian ke file direct.csv.

3.1.4.3. Hasil Pengamatan

3.1.4.3.1. Client

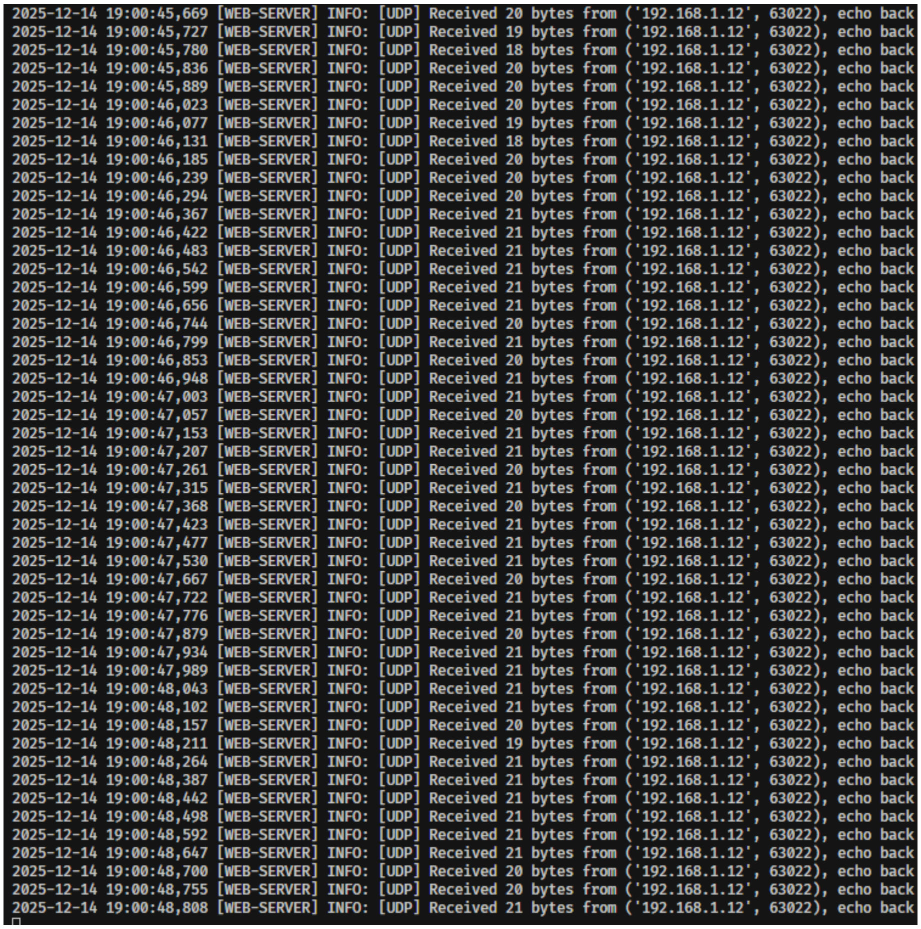


```
Pilih menu: 4
UDP Direct
2025-12-14 19:00:44,914 [CLIENT] INFO: Seq 1 RTT 8.12 ms
2025-12-14 19:00:44,969 [CLIENT] INFO: Seq 2 RTT 63.43 ms
2025-12-14 19:00:45,023 [CLIENT] INFO: Seq 3 RTT 116.70 ms
2025-12-14 19:00:45,079 [CLIENT] INFO: Seq 4 RTT 173.24 ms
2025-12-14 19:00:45,132 [CLIENT] INFO: Seq 5 RTT 225.84 ms
2025-12-14 19:00:45,265 [CLIENT] INFO: Seq 6 RTT 359.65 ms
2025-12-14 19:00:45,319 [CLIENT] INFO: Seq 7 RTT 413.67 ms
2025-12-14 19:00:45,374 [CLIENT] INFO: Seq 8 RTT 467.84 ms
2025-12-14 19:00:45,428 [CLIENT] INFO: Seq 9 RTT 521.82 ms
2025-12-14 19:00:45,483 [CLIENT] INFO: Seq 10 RTT 577.38 ms
2025-12-14 19:00:45,537 [CLIENT] INFO: Seq 11 RTT 631.05 ms
2025-12-14 19:00:45,610 [CLIENT] INFO: Seq 12 RTT 704.27 ms
2025-12-14 19:00:45,665 [CLIENT] INFO: Seq 13 RTT 759.30 ms
2025-12-14 19:00:45,727 [CLIENT] INFO: Seq 14 RTT 820.93 ms
2025-12-14 19:00:45,788 [CLIENT] INFO: Seq 15 RTT 882.15 ms
2025-12-14 19:00:45,842 [CLIENT] INFO: Seq 16 RTT 936.25 ms
2025-12-14 19:00:45,899 [CLIENT] INFO: Seq 17 RTT 993.34 ms
2025-12-14 19:00:45,987 [CLIENT] INFO: Seq 18 RTT 1080.70 ms
2025-12-14 19:00:46,042 [CLIENT] INFO: Seq 19 RTT 1136.53 ms
2025-12-14 19:00:46,096 [CLIENT] INFO: Seq 20 RTT 1190.12 ms
2025-12-14 19:00:46,192 [CLIENT] INFO: Seq 21 RTT 1286.29 ms
2025-12-14 19:00:46,246 [CLIENT] INFO: Seq 22 RTT 1339.70 ms
2025-12-14 19:00:46,302 [CLIENT] INFO: Seq 23 RTT 1396.22 ms
2025-12-14 19:00:46,396 [CLIENT] INFO: Seq 24 RTT 1490.52 ms
2025-12-14 19:00:46,450 [CLIENT] INFO: Seq 25 RTT 1544.46 ms
2025-12-14 19:00:46,505 [CLIENT] INFO: Seq 26 RTT 1598.71 ms
2025-12-14 19:00:46,558 [CLIENT] INFO: Seq 27 RTT 1652.02 ms
2025-12-14 19:00:46,613 [CLIENT] INFO: Seq 28 RTT 1706.77 ms
2025-12-14 19:00:46,666 [CLIENT] INFO: Seq 29 RTT 1760.01 ms
2025-12-14 19:00:46,719 [CLIENT] INFO: Seq 30 RTT 1813.27 ms
2025-12-14 19:00:46,772 [CLIENT] INFO: Seq 31 RTT 1866.33 ms
2025-12-14 19:00:46,911 [CLIENT] INFO: Seq 32 RTT 2004.95 ms
2025-12-14 19:00:46,965 [CLIENT] INFO: Seq 33 RTT 2059.30 ms
2025-12-14 19:00:47,019 [CLIENT] INFO: Seq 34 RTT 2112.71 ms
2025-12-14 19:00:47,122 [CLIENT] INFO: Seq 35 RTT 2215.69 ms
2025-12-14 19:00:47,177 [CLIENT] INFO: Seq 36 RTT 2271.45 ms
2025-12-14 19:00:47,233 [CLIENT] INFO: Seq 37 RTT 2326.83 ms
2025-12-14 19:00:47,287 [CLIENT] INFO: Seq 38 RTT 2380.77 ms
2025-12-14 19:00:47,345 [CLIENT] INFO: Seq 39 RTT 2439.19 ms
2025-12-14 19:00:47,400 [CLIENT] INFO: Seq 40 RTT 2494.19 ms
2025-12-14 19:00:47,453 [CLIENT] INFO: Seq 41 RTT 2547.65 ms
2025-12-14 19:00:47,508 [CLIENT] INFO: Seq 42 RTT 2602.13 ms
2025-12-14 19:00:47,631 [CLIENT] INFO: Seq 43 RTT 2725.09 ms
2025-12-14 19:00:47,686 [CLIENT] INFO: Seq 44 RTT 2780.16 ms
2025-12-14 19:00:47,743 [CLIENT] INFO: Seq 45 RTT 2836.91 ms
2025-12-14 19:00:47,836 [CLIENT] INFO: Seq 46 RTT 2929.97 ms
2025-12-14 19:00:47,890 [CLIENT] INFO: Seq 47 RTT 2983.83 ms
2025-12-14 19:00:47,945 [CLIENT] INFO: Seq 48 RTT 3038.84 ms
2025-12-14 19:00:47,997 [CLIENT] INFO: Seq 49 RTT 3091.47 ms
2025-12-14 19:00:48,051 [CLIENT] INFO: Seq 50 RTT 3144.75 ms
2025-12-14 19:00:48,103 [CLIENT] INFO: CSV disimpan: direct.csv
```

Pada terminal client terlihat log pengiriman dan penerimaan paket UDP dengan penanda Seq dan nilai RTT masing-masing paket. RTT menunjukkan waktu bolak-balik dari client ke web server dan kembali ke client. Di akhir pengujian, client menyimpan data RTT ke file direct.csv

Nilai RTT pada pengujian ini terlihat bervariasi dan cenderung meningkat seiring waktu, yang menunjukkan adanya fluktuasi delay jaringan. Hal ini wajar pada UDP karena tidak ada mekanisme kontrol kongesti atau retransmisi seperti pada TCP.

3.1.4.3.2. Web Server



```
2025-12-14 19:00:45,669 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:45,727 [WEB-SERVER] INFO: [UDP] Received 19 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:45,780 [WEB-SERVER] INFO: [UDP] Received 18 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:45,836 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:45,889 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,023 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,077 [WEB-SERVER] INFO: [UDP] Received 19 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,131 [WEB-SERVER] INFO: [UDP] Received 18 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,185 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,239 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,294 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,367 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,422 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,483 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,542 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,599 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,656 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,744 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,799 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,853 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:46,948 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,003 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,057 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,153 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,207 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,261 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,315 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,368 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,423 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,477 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,530 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,667 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,722 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,776 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,879 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,934 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:47,989 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,043 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,102 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,157 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,211 [WEB-SERVER] INFO: [UDP] Received 19 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,264 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,387 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,442 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,498 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,592 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,647 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,700 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,755 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.12', 63022), echo back
2025-12-14 19:00:48,808 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.12', 63022), echo back
```

Pada terminal web server terlihat log [UDP] Received ... echo back yang menandakan bahwa setiap paket UDP dari client berhasil diterima dan langsung dikirim kembali. Log ini membuktikan bahwa fungsi UDP echo server berjalan dengan benar dan komunikasi direct antara client dan web server berlangsung tanpa perantara.

3.1.4.3.3. Proxy Server

Pada pengujian UDP direct, tidak ada log yang muncul di terminal proxy server. Hal ini normal dan sesuai dengan desain sistem, karena client mengirim paket UDP langsung ke port UDP web server (9000) tanpa melewati proxy. Proxy server hanya akan aktif dan mencatat log apabila client mengirim paket ke port proxy UDP (9090).

3.1.4.3.4. Kesimpulan

Pengujian UDP direct menunjukkan bahwa komunikasi UDP antara client dan web server dapat berjalan dengan baik tanpa proxy. Web server berhasil menerima dan mengembalikan paket echo, sementara client mampu mengukur RTT dan menyimpan hasil pengujian ke file CSV. Tidak adanya log pada proxy server menegaskan bahwa jalur komunikasi memang langsung (direct) tanpa perantara. Hasil ini menjadi baseline performa jaringan yang penting untuk dibandingkan dengan pengujian UDP melalui proxy pada tahap selanjutnya.

3.1.5. Pengujian UDP Via Proxy

3.1.5.1. Tujuan Pengujian

Pengujian UDP via proxy dilakukan untuk mengetahui dampak penggunaan proxy server terhadap kualitas layanan komunikasi UDP. Fokus pengujian ini adalah mengamati perubahan nilai RTT (Round Trip Time), packet loss, jitter, dan throughput ketika paket UDP dari client tidak langsung menuju web server, tetapi harus melewati proxy server terlebih dahulu. Selain itu, pengujian ini bertujuan memastikan bahwa proxy server mampu meneruskan paket UDP dengan benar dan stabil.

3.1.5.2. Proses Pengujian

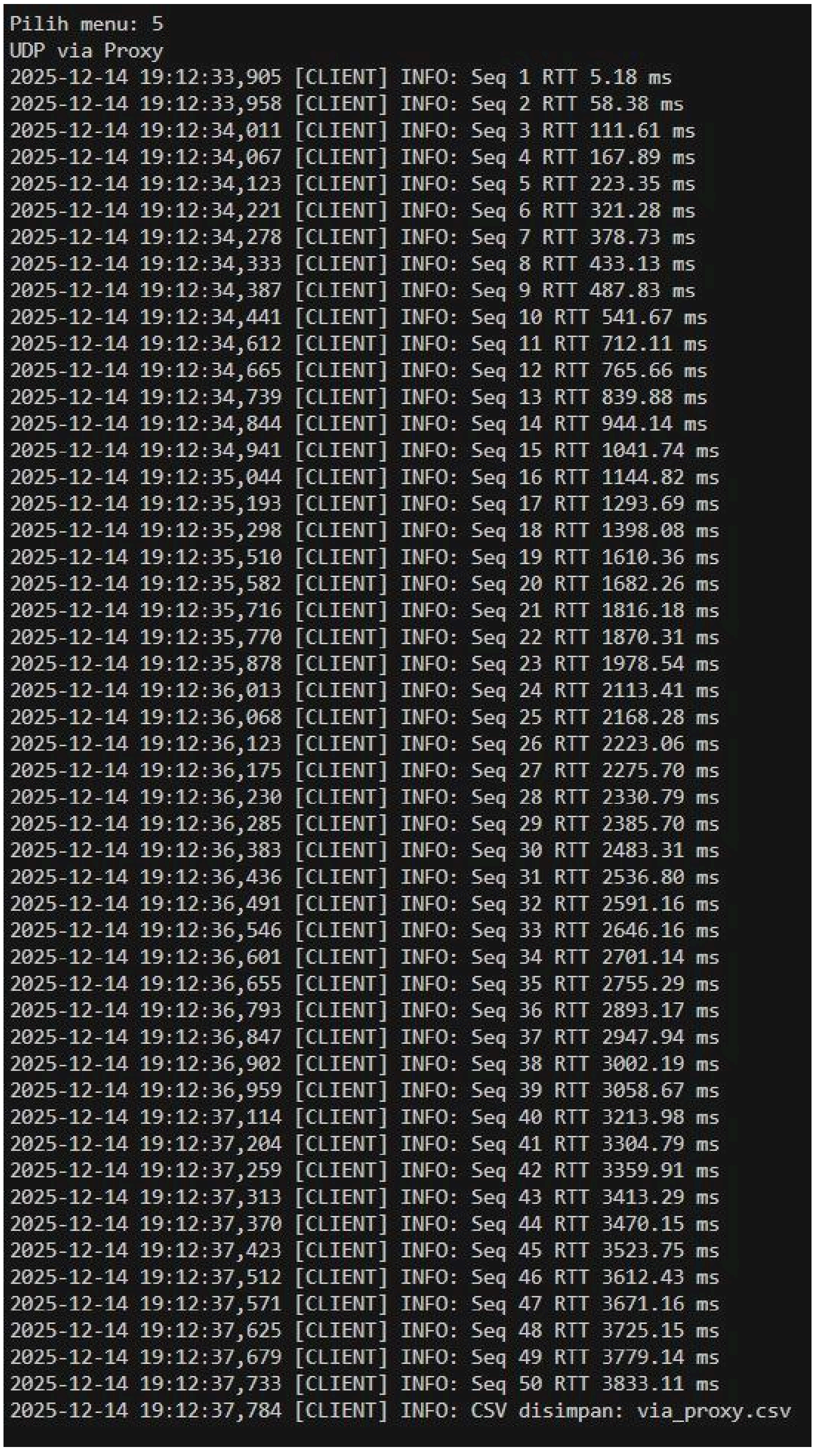
Proses pengujian UDP via proxy dilakukan dengan langkah-langkah sebagai berikut:

1. Web server dijalankan pada Laptop A dengan UDP echo server aktif di port 9000.
2. Proxy server dijalankan pada Laptop A yang mendengarkan koneksi UDP di port 9090 dan meneruskan paket ke web server.

3. Client dijalankan pada Laptop B dan memilih menu UDP via Proxy.
4. Client mengirimkan 50 paket UDP berukuran tetap secara berurutan ke proxy server.
5. Proxy server meneruskan paket ke web server dan mengembalikan respons echo ke client.
6. Client mencatat RTT setiap paket dan menyimpan hasilnya ke file via_proxy.csv.

3.1.5.3. Hasil Pengamatan

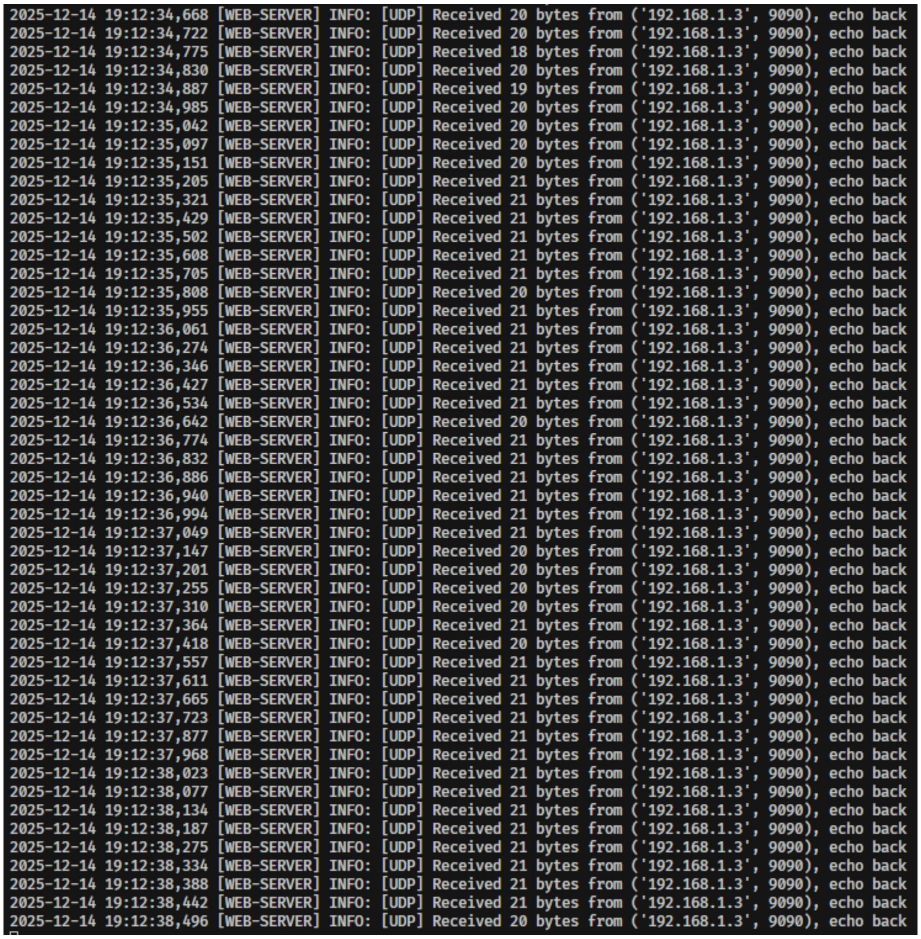
3.1.5.3.1. Client



```
Pilih menu: 5
UDP via Proxy
2025-12-14 19:12:33,905 [CLIENT] INFO: Seq 1 RTT 5.18 ms
2025-12-14 19:12:33,958 [CLIENT] INFO: Seq 2 RTT 58.38 ms
2025-12-14 19:12:34,011 [CLIENT] INFO: Seq 3 RTT 111.61 ms
2025-12-14 19:12:34,067 [CLIENT] INFO: Seq 4 RTT 167.89 ms
2025-12-14 19:12:34,123 [CLIENT] INFO: Seq 5 RTT 223.35 ms
2025-12-14 19:12:34,221 [CLIENT] INFO: Seq 6 RTT 321.28 ms
2025-12-14 19:12:34,278 [CLIENT] INFO: Seq 7 RTT 378.73 ms
2025-12-14 19:12:34,333 [CLIENT] INFO: Seq 8 RTT 433.13 ms
2025-12-14 19:12:34,387 [CLIENT] INFO: Seq 9 RTT 487.83 ms
2025-12-14 19:12:34,441 [CLIENT] INFO: Seq 10 RTT 541.67 ms
2025-12-14 19:12:34,612 [CLIENT] INFO: Seq 11 RTT 712.11 ms
2025-12-14 19:12:34,665 [CLIENT] INFO: Seq 12 RTT 765.66 ms
2025-12-14 19:12:34,739 [CLIENT] INFO: Seq 13 RTT 839.88 ms
2025-12-14 19:12:34,844 [CLIENT] INFO: Seq 14 RTT 944.14 ms
2025-12-14 19:12:34,941 [CLIENT] INFO: Seq 15 RTT 1041.74 ms
2025-12-14 19:12:35,044 [CLIENT] INFO: Seq 16 RTT 1144.82 ms
2025-12-14 19:12:35,193 [CLIENT] INFO: Seq 17 RTT 1293.69 ms
2025-12-14 19:12:35,298 [CLIENT] INFO: Seq 18 RTT 1398.08 ms
2025-12-14 19:12:35,510 [CLIENT] INFO: Seq 19 RTT 1610.36 ms
2025-12-14 19:12:35,582 [CLIENT] INFO: Seq 20 RTT 1682.26 ms
2025-12-14 19:12:35,716 [CLIENT] INFO: Seq 21 RTT 1816.18 ms
2025-12-14 19:12:35,770 [CLIENT] INFO: Seq 22 RTT 1870.31 ms
2025-12-14 19:12:35,878 [CLIENT] INFO: Seq 23 RTT 1978.54 ms
2025-12-14 19:12:36,013 [CLIENT] INFO: Seq 24 RTT 2113.41 ms
2025-12-14 19:12:36,068 [CLIENT] INFO: Seq 25 RTT 2168.28 ms
2025-12-14 19:12:36,123 [CLIENT] INFO: Seq 26 RTT 2223.06 ms
2025-12-14 19:12:36,175 [CLIENT] INFO: Seq 27 RTT 2275.70 ms
2025-12-14 19:12:36,230 [CLIENT] INFO: Seq 28 RTT 2330.79 ms
2025-12-14 19:12:36,285 [CLIENT] INFO: Seq 29 RTT 2385.70 ms
2025-12-14 19:12:36,383 [CLIENT] INFO: Seq 30 RTT 2483.31 ms
2025-12-14 19:12:36,436 [CLIENT] INFO: Seq 31 RTT 2536.80 ms
2025-12-14 19:12:36,491 [CLIENT] INFO: Seq 32 RTT 2591.16 ms
2025-12-14 19:12:36,546 [CLIENT] INFO: Seq 33 RTT 2646.16 ms
2025-12-14 19:12:36,601 [CLIENT] INFO: Seq 34 RTT 2701.14 ms
2025-12-14 19:12:36,655 [CLIENT] INFO: Seq 35 RTT 2755.29 ms
2025-12-14 19:12:36,793 [CLIENT] INFO: Seq 36 RTT 2893.17 ms
2025-12-14 19:12:36,847 [CLIENT] INFO: Seq 37 RTT 2947.94 ms
2025-12-14 19:12:36,902 [CLIENT] INFO: Seq 38 RTT 3002.19 ms
2025-12-14 19:12:36,959 [CLIENT] INFO: Seq 39 RTT 3058.67 ms
2025-12-14 19:12:37,114 [CLIENT] INFO: Seq 40 RTT 3213.98 ms
2025-12-14 19:12:37,204 [CLIENT] INFO: Seq 41 RTT 3304.79 ms
2025-12-14 19:12:37,259 [CLIENT] INFO: Seq 42 RTT 3359.91 ms
2025-12-14 19:12:37,313 [CLIENT] INFO: Seq 43 RTT 3413.29 ms
2025-12-14 19:12:37,370 [CLIENT] INFO: Seq 44 RTT 3470.15 ms
2025-12-14 19:12:37,423 [CLIENT] INFO: Seq 45 RTT 3523.75 ms
2025-12-14 19:12:37,512 [CLIENT] INFO: Seq 46 RTT 3612.43 ms
2025-12-14 19:12:37,571 [CLIENT] INFO: Seq 47 RTT 3671.16 ms
2025-12-14 19:12:37,625 [CLIENT] INFO: Seq 48 RTT 3725.15 ms
2025-12-14 19:12:37,679 [CLIENT] INFO: Seq 49 RTT 3779.14 ms
2025-12-14 19:12:37,733 [CLIENT] INFO: Seq 50 RTT 3833.11 ms
2025-12-14 19:12:37,784 [CLIENT] INFO: CSV disimpan: via_proxy.csv
```

Pada terminal client terlihat bahwa setiap paket UDP yang dikirim menghasilkan nilai RTT yang tercatat. RTT meningkat secara bertahap seiring bertambahnya nomor urutan paket. Setelah seluruh paket dikirim, client menampilkan informasi bahwa file CSV (via_proxy.csv) berhasil disimpan. Hal ini menunjukkan bahwa client berhasil melakukan pengujian UDP melalui proxy dan mengumpulkan data QoS secara lengkap.

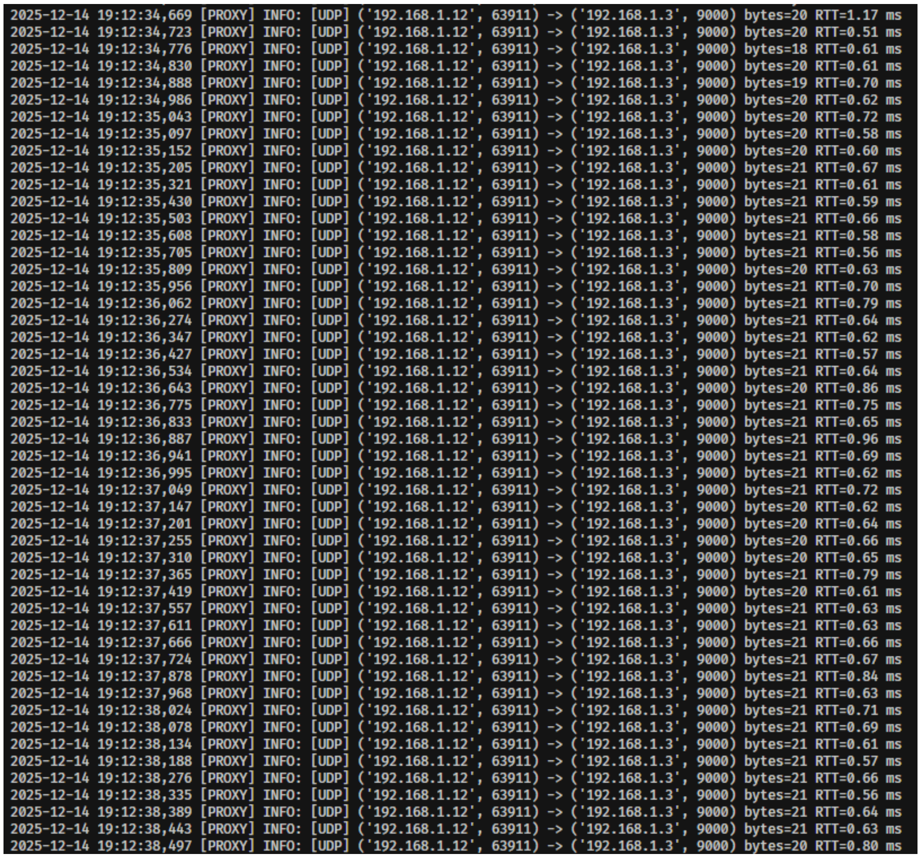
3.1.5.3.2. Web Server



```
2025-12-14 19:12:34,668 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:34,722 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:34,775 [WEB-SERVER] INFO: [UDP] Received 18 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:34,830 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:34,887 [WEB-SERVER] INFO: [UDP] Received 19 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:34,985 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:35,042 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:35,097 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:35,151 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:35,205 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:35,321 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:35,429 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:35,502 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:35,608 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:35,705 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:35,808 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:35,955 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:36,061 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:36,274 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:36,346 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:36,427 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:36,534 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:36,642 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:36,774 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:36,832 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:36,886 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:36,940 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:36,994 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,049 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,147 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,201 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,255 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,310 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,364 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,418 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,557 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,611 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,665 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,723 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,877 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:37,968 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:38,023 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:38,077 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:38,134 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:38,187 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:38,275 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:38,334 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:38,388 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:38,442 [WEB-SERVER] INFO: [UDP] Received 21 bytes from ('192.168.1.3', 9090), echo back
2025-12-14 19:12:38,496 [WEB-SERVER] INFO: [UDP] Received 20 bytes from ('192.168.1.3', 9090), echo back
```

Terminal web server menampilkan log penerimaan paket UDP dari alamat IP proxy server, bukan langsung dari client. Setiap paket yang diterima langsung dikirim kembali sebagai echo. Hal ini menunjukkan bahwa web server hanya berinteraksi dengan proxy, sehingga peran proxy sebagai perantara berjalan sesuai desain.

3.1.5.3.3. Proxy server



```
2025-12-14 19:12:34,669 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=1.17 ms
2025-12-14 19:12:34,723 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.51 ms
2025-12-14 19:12:34,776 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=18 RTT=0.61 ms
2025-12-14 19:12:34,830 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.61 ms
2025-12-14 19:12:34,888 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=19 RTT=0.70 ms
2025-12-14 19:12:34,986 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.62 ms
2025-12-14 19:12:35,043 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.72 ms
2025-12-14 19:12:35,097 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.58 ms
2025-12-14 19:12:35,152 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.60 ms
2025-12-14 19:12:35,205 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.67 ms
2025-12-14 19:12:35,321 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.61 ms
2025-12-14 19:12:35,430 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.59 ms
2025-12-14 19:12:35,503 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.66 ms
2025-12-14 19:12:35,608 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.58 ms
2025-12-14 19:12:35,705 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.56 ms
2025-12-14 19:12:35,809 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.63 ms
2025-12-14 19:12:35,956 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.70 ms
2025-12-14 19:12:36,062 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.79 ms
2025-12-14 19:12:36,274 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.64 ms
2025-12-14 19:12:36,347 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.62 ms
2025-12-14 19:12:36,427 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.57 ms
2025-12-14 19:12:36,534 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.64 ms
2025-12-14 19:12:36,643 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.86 ms
2025-12-14 19:12:36,775 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.75 ms
2025-12-14 19:12:36,833 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.65 ms
2025-12-14 19:12:36,887 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.96 ms
2025-12-14 19:12:36,941 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.69 ms
2025-12-14 19:12:36,995 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.62 ms
2025-12-14 19:12:37,049 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.72 ms
2025-12-14 19:12:37,147 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.62 ms
2025-12-14 19:12:37,201 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.64 ms
2025-12-14 19:12:37,255 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.66 ms
2025-12-14 19:12:37,310 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.65 ms
2025-12-14 19:12:37,365 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.79 ms
2025-12-14 19:12:37,419 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.61 ms
2025-12-14 19:12:37,557 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.63 ms
2025-12-14 19:12:37,611 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.63 ms
2025-12-14 19:12:37,666 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.66 ms
2025-12-14 19:12:37,724 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.67 ms
2025-12-14 19:12:37,878 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.84 ms
2025-12-14 19:12:37,968 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.63 ms
2025-12-14 19:12:38,024 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.71 ms
2025-12-14 19:12:38,078 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.69 ms
2025-12-14 19:12:38,134 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.61 ms
2025-12-14 19:12:38,188 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.57 ms
2025-12-14 19:12:38,276 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.66 ms
2025-12-14 19:12:38,335 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.56 ms
2025-12-14 19:12:38,389 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.64 ms
2025-12-14 19:12:38,443 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=21 RTT=0.63 ms
2025-12-14 19:12:38,497 [PROXY] INFO: [UDP] ('192.168.1.12', 63911) -> ('192.168.1.3', 9000) bytes=20 RTT=0.80 ms
```

Terminal proxy server menampilkan log detail setiap paket UDP yang diterima dari client dan diteruskan ke web server. Pada log ini terlihat informasi ukuran paket dan RTT versi proxy yang relatif kecil dan stabil. Hal ini menandakan bahwa proses forwarding di proxy berjalan cepat, dan sebagian besar tambahan delay berasal dari antrean atau interval pengiriman paket di sisi client.

3.1.5.3.4. Kesimpulan

Penggunaan proxy pada komunikasi UDP menambahkan satu lapisan perantara yang memengaruhi performa jaringan. Nilai RTT yang diterima client cenderung lebih besar dibandingkan pengujian UDP direct karena paket harus melewati proxy sebelum mencapai web server. Namun demikian, proxy server mampu meneruskan paket dengan konsisten tanpa packet loss yang signifikan. Pengujian ini membuktikan bahwa proxy server dapat digunakan untuk monitoring dan kontrol lalu lintas UDP, meskipun dengan konsekuensi penambahan delay pada komunikasi end-to-end.

3.1.6. Wireshark Capture

3.1.6.1. Tujuan Pengujian

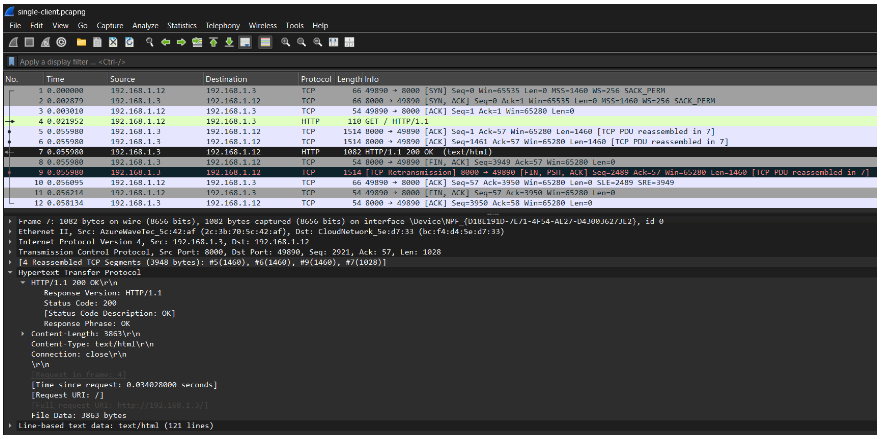
Pengujian Wireshark Capture dilakukan untuk mengamati dan menganalisis lalu lintas jaringan yang terjadi selama proses komunikasi antara client, proxy server, dan web server. Tujuan utama dari pengujian ini adalah untuk memverifikasi proses pertukaran data pada jaringan, mengamati pembentukan dan terminasi koneksi, serta mendukung analisis Quality of Service (QoS) seperti latency, throughput, packet loss, dan jitter berdasarkan paket yang tertangkap.

3.1.6.2. Proses Pengujian

Proses pengujian dilakukan dengan memastikan seluruh perangkat yang digunakan, yaitu client, proxy server, dan web server, terhubung pada jaringan WiFi yang sama. Setelah itu, aplikasi Wireshark dijalankan pada laptop client untuk menangkap paket jaringan. Interface jaringan yang aktif dipilih, kemudian proses capture dimulai sebelum client menjalankan pengujian

3.1.6.3. Hasil Pengamatan

3.1.6.3.1. Single-client.pcapng


single-client.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time      Source        Destination   Protocol Length Info
    1 0.000000  192.168.1.12  192.168.1.3   TCP        66 49890 → 8000 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    2 0.002879  192.168.1.3   192.168.1.12  TCP        66 8000 → 49890 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    3 0.003010  192.168.1.12  192.168.1.3   TCP        54 49890 → 8000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
    4 0.021952  192.168.1.12  192.168.1.3   HTTP      110 GET / HTTP/1.1
    5 0.055980  192.168.1.3   192.168.1.12  TCP      1514 8000 → 49890 [ACK] Seq=1 Ack=57 Win=65280 Len=1460 [TCP PDU reassembled in 7]
    6 0.055980  192.168.1.3   192.168.1.12  TCP      1514 8000 → 49890 [ACK] Seq=1461 Ack=57 Win=65280 Len=1460 [TCP PDU reassembled in 7]
    7 0.055980  192.168.1.3   192.168.1.12  HTTP     1082 HTTP/1.1 200 OK  (text/html)
    8 0.055980  192.168.1.3   192.168.1.12  TCP        54 8000 → 49890 [FIN, ACK] Seq=3949 Ack=57 Win=65280 Len=0
    9 0.055980  192.168.1.3   192.168.1.12  TCP      1514 [TCP Retransmission] 8000 → 49890 [FIN, PSH, ACK] Seq=2489 Ack=57 Win=65280 Len=1460 [TCP PDU reassembled in 7]
   10 0.056095  192.168.1.12  192.168.1.3   TCP        66 49890 → 8000 [ACK] Seq=57 Ack=3950 Win=65280 Len=0 SLE=2489 SRE=3949
   11 0.056214  192.168.1.12  192.168.1.3   TCP        54 49890 → 8000 [FIN, ACK] Seq=57 Ack=3950 Win=65280 Len=0
   12 0.058134  192.168.1.3   192.168.1.12  TCP        54 8000 → 49890 [ACK] Seq=3950 Ack=58 Win=65280 Len=0
```

```
> Frame 7: 1082 bytes on wire (8656 bits), 1082 bytes captured (8656 bits) on interface \Device\NPF_{D18E191D-7E71-4F54-AE27-D430036273E2}, id 0
> Ethernet II, Src: AzureWaveTec_5c:42:af (2c:3b:70:5c:42:af), Dst: CloudNetwork_5e:d7:33 (bc:f4:d4:5e:d7:33)
> Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.12
> Transmission Control Protocol, Src Port: 8000, Dst Port: 49890, Seq: 2921, Ack: 57, Len: 1028
> [4 Reassembled TCP Segments (3948 bytes): #5(1460), #6(1460), #9(1460), #7(1028)]
v Hypertext Transfer Protocol
  v HTTP/1.1 200 OK\r\n
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
  > Content-Length: 3863\r\n
    Content-Type: text/html\r\n
    Connection: close\r\n
    \r\n
    [Request in frame: 4]
    [Time since request: 0.034028000 seconds]
    [Request URI: /]
    [Full request URI: http://192.168.1.3/]
    File Data: 3863 bytes
> Line-based text data: text/html (121 lines)
```

Frame	Total Data Dikirim	Waktu Pengiriman	Throughput
4-7	880+8656 bits = 9536 bits	0.034028000 seconds	9596 / 0.03408 ≈ 280,18 bps

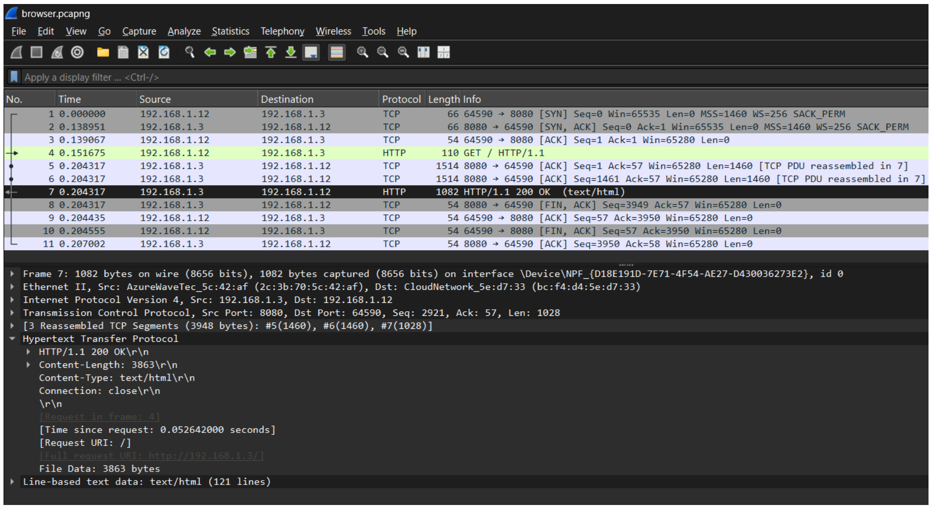
Latency = Time Response−Time Request

0.034028000 - 0.021952000 = 0.012076 / 34.028 ms

Tidak ada packet loss atau packet 0%

Jitter =

3.1.6.3.2. browser[1].pcapng


browser.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time      Source        Destination   Protocol Length Info
   1 0.000000  192.168.1.12  192.168.1.3   TCP        66 64590 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
   2 0.138951  192.168.1.3   192.168.1.12  TCP        66 8080 → 64590 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
   3 0.139067  192.168.1.12  192.168.1.3   TCP        54 64590 → 8080 [ACK] Seq=1 Ack=1 Win=65280 Len=0
   4 0.151675  192.168.1.12  192.168.1.3   HTTP      110 GET / HTTP/1.1
   5 0.204317  192.168.1.3   192.168.1.12  TCP      1514 8080 → 64590 [ACK] Seq=1 Ack=57 Win=65280 Len=1460 [TCP PDU reassembled in 7]
   6 0.204317  192.168.1.3   192.168.1.12  TCP      1514 8080 → 64590 [ACK] Seq=1461 Ack=57 Win=65280 Len=1460 [TCP PDU reassembled in 7]
   7 0.204317  192.168.1.3   192.168.1.12  HTTP     1082 HTTP/1.1 200 OK  (text/html)
   8 0.204317  192.168.1.3   192.168.1.12  TCP        54 8080 → 64590 [FIN, ACK] Seq=3949 Ack=57 Win=65280 Len=0
   9 0.204435  192.168.1.12  192.168.1.3   TCP        54 64590 → 8080 [ACK] Seq=57 Ack=3950 Win=65280 Len=0
  10 0.204555  192.168.1.12  192.168.1.3   TCP        54 64590 → 8080 [FIN, ACK] Seq=57 Ack=3950 Win=65280 Len=0
  11 0.207002  192.168.1.3   192.168.1.12  TCP        54 8080 → 64590 [ACK] Seq=3950 Ack=58 Win=65280 Len=0

> Frame 7: 1082 bytes on wire (8656 bits), 1082 bytes captured (8656 bits) on interface \Device\NPF_{D18E191D-7E71-4F54-AE27-D430036273E2}, id 0
> Ethernet II, Src: AzureWaveTec_5c:42:af (2c:3b:70:5c:42:af), Dst: CloudNetwork_5e:d7:33 (bc:f4:d4:5e:d7:33)
> Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.12
> Transmission Control Protocol, Src Port: 8080, Dst Port: 64590, Seq: 2921, Ack: 57, Len: 1028
> [3 Reassembled TCP Segments (3948 bytes): #5(1460), #6(1460), #7(1028)]
v Hypertext Transfer Protocol
  > HTTP/1.1 200 OK\r\n
  > Content-Length: 3863\r\n
    Content-Type: text/html\r\n
    Connection: close\r\n
    \r\n
    [Request in frame: 4]
    [Time since request: 0.052642000 seconds]
    [Request URI: /]
    [Full request URI: http://192.168.1.3/]
    File Data: 3863 bytes
> Line-based text data: text/html (121 lines)
```

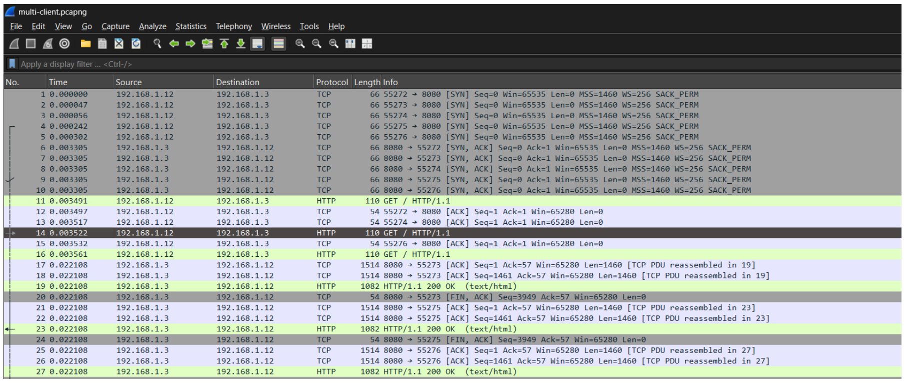
Frame	Total Data Dikirim	Waktu Pengiriman	throughput
4-7	3863 x 8 = 31704 bits	0.052642000	30904/0.052642000 = 587059.76 bps

Latency: Waktu Penerimaan Paket - Waktu Pengiriman Paket = 0.052642000 \|\| 52.642 ms

Packet loss 0%

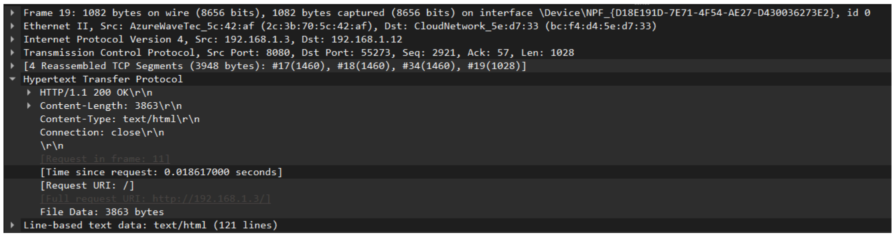
Jitter: 0 ms

3.1.6.3.3. Multi-client[1].pcapng


multi-client.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

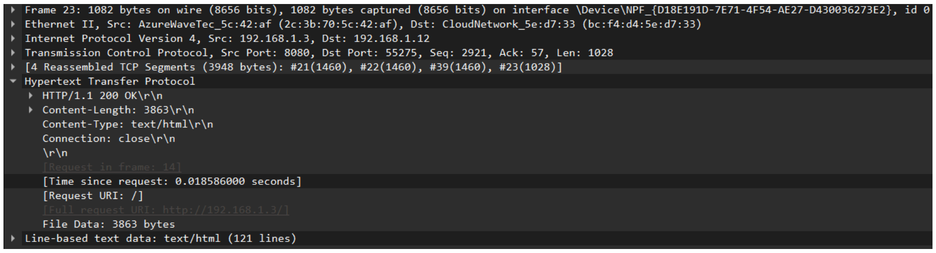
```
No.   Time       Source         Destination    Protocol Length Info
    1 0.000000   192.168.1.12   192.168.1.3    TCP        66 55272 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    2 0.000047   192.168.1.12   192.168.1.3    TCP        66 55273 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    3 0.000056   192.168.1.12   192.168.1.3    TCP        66 55274 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    4 0.000242   192.168.1.12   192.168.1.3    TCP        66 55275 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    5 0.000302   192.168.1.12   192.168.1.3    TCP        66 55276 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    6 0.003305   192.168.1.3    192.168.1.12   TCP        66 8080 → 55272 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    7 0.003305   192.168.1.3    192.168.1.12   TCP        66 8080 → 55273 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    8 0.003305   192.168.1.3    192.168.1.12   TCP        66 8080 → 55274 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
    9 0.003305   192.168.1.3    192.168.1.12   TCP        66 8080 → 55275 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
   10 0.003305   192.168.1.3    192.168.1.12   TCP        66 8080 → 55276 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
   11 0.003491   192.168.1.12   192.168.1.3    HTTP      110 GET / HTTP/1.1
   12 0.003497   192.168.1.12   192.168.1.3    TCP        54 55272 → 8080 [ACK] Seq=1 Ack=1 Win=65280 Len=0
   13 0.003517   192.168.1.12   192.168.1.3    TCP        54 55274 → 8080 [ACK] Seq=1 Ack=1 Win=65280 Len=0
   14 0.003522   192.168.1.12   192.168.1.3    HTTP      110 GET / HTTP/1.1
   15 0.003532   192.168.1.12   192.168.1.3    TCP        54 55276 → 8080 [ACK] Seq=1 Ack=1 Win=65280 Len=0
   16 0.003561   192.168.1.12   192.168.1.3    HTTP      110 GET / HTTP/1.1
   17 0.022108   192.168.1.3    192.168.1.12   TCP      1514 8080 → 55273 [ACK] Seq=1 Ack=57 Win=65280 Len=1460 [TCP PDU reassembled in 19]
   18 0.022108   192.168.1.3    192.168.1.12   TCP      1514 8080 → 55273 [ACK] Seq=1461 Ack=57 Win=65280 Len=1460 [TCP PDU reassembled in 19]
   19 0.022108   192.168.1.3    192.168.1.12   HTTP     1082 HTTP/1.1 200 OK  (text/html)
   20 0.022108   192.168.1.3    192.168.1.12   TCP        54 8080 → 55273 [FIN, ACK] Seq=3949 Ack=57 Win=65280 Len=0
   21 0.022108   192.168.1.3    192.168.1.12   TCP      1514 8080 → 55275 [ACK] Seq=1 Ack=57 Win=65280 Len=1460 [TCP PDU reassembled in 23]
   22 0.022108   192.168.1.3    192.168.1.12   TCP      1514 8080 → 55275 [ACK] Seq=1461 Ack=57 Win=65280 Len=1460 [TCP PDU reassembled in 23]
   23 0.022108   192.168.1.3    192.168.1.12   HTTP     1082 HTTP/1.1 200 OK  (text/html)
   24 0.022108   192.168.1.3    192.168.1.12   TCP        54 8080 → 55275 [FIN, ACK] Seq=3949 Ack=57 Win=65280 Len=0
   25 0.022108   192.168.1.3    192.168.1.12   TCP      1514 8080 → 55276 [ACK] Seq=1 Ack=57 Win=65280 Len=1460 [TCP PDU reassembled in 27]
   26 0.022108   192.168.1.3    192.168.1.12   TCP      1514 8080 → 55276 [ACK] Seq=1461 Ack=57 Win=65280 Len=1460 [TCP PDU reassembled in 27]
   27 0.022108   192.168.1.3    192.168.1.12   HTTP     1082 HTTP/1.1 200 OK  (text/html)
```

Client 1 (frame 11 & 19)



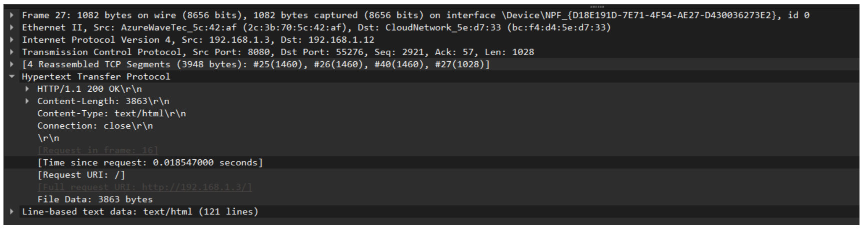
```
Frame 19: 1082 bytes on wire (8656 bits), 1082 bytes captured (8656 bits) on interface \Device\NPF_{D18E191D-7E71-4F54-AE27-D430036273E2}, id 0
Ethernet II, Src: AzureWaveTec_5c:42:af (2c:3b:70:5c:42:af), Dst: CloudNetwork_5e:d7:33 (bc:f4:d4:5e:d7:33)
Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.12
Transmission Control Protocol, Src Port: 8080, Dst Port: 55273, Seq: 2921, Ack: 57, Len: 1028
[4 Reassembled TCP Segments (3948 bytes): #17(1460), #18(1460), #34(1460), #19(1028)]
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Content-Length: 3863\r\n
  Content-Type: text/html\r\n
  Connection: close\r\n
  \r\n
  [Request in frame: 11]
  [Time since request: 0.018617000 seconds]
  [Request URI: /]
  [Full request URI: http://192.168.1.3/]
  File Data: 3863 bytes
Line-based text data: text/html (121 lines)
```

Client 2 (Frame 14 & 23)



```
Frame 23: 1082 bytes on wire (8656 bits), 1082 bytes captured (8656 bits) on interface \Device\NPF_{D18E191D-7E71-4F54-AE27-D430036273E2}, id 0
Ethernet II, Src: AzureWaveTec_5c:42:af (2c:3b:70:5c:42:af), Dst: CloudNetwork_5e:d7:33 (bc:f4:d4:5e:d7:33)
Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.12
Transmission Control Protocol, Src Port: 8080, Dst Port: 55275, Seq: 2921, Ack: 57, Len: 1028
[4 Reassembled TCP Segments (3948 bytes): #21(1460), #22(1460), #39(1460), #23(1028)]
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Content-Length: 3863\r\n
  Content-Type: text/html\r\n
  Connection: close\r\n
  \r\n
  [Request in frame: 14]
  [Time since request: 0.018586000 seconds]
  [Request URI: /]
  [Full request URI: http://192.168.1.3/]
  File Data: 3863 bytes
Line-based text data: text/html (121 lines)
```

Client 3 (Frame 16 & 27)



```
Frame 27: 1082 bytes on wire (8656 bits), 1082 bytes captured (8656 bits) on interface \Device\NPF_{D18E191D-7E71-4F54-AE27-D430036273E2}, id 0
Ethernet II, Src: AzureWaveTec_5c:42:af (2c:3b:70:5c:42:af), Dst: CloudNetwork_5e:d7:33 (bc:f4:d4:5e:d7:33)
Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.12
Transmission Control Protocol, Src Port: 8080, Dst Port: 55276, Seq: 2921, Ack: 57, Len: 1028
[4 Reassembled TCP Segments (3948 bytes): #25(1460), #26(1460), #40(1460), #27(1028)]
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Content-Length: 3863\r\n
  Content-Type: text/html\r\n
  Connection: close\r\n
  \r\n
  [Request in frame: 16]
  [Time since request: 0.018547000 seconds]
  [Request URI: /]
  [Full request URI: http://192.168.1.3/]
  File Data: 3863 bytes
Line-based text data: text/html (121 lines)
```

Latency = 0.018617000 s (client 1)

Latency = 0.018586000 s (client 2)

Latency = 0.018547000 s (client 3)

Latency rata-rata = $\frac{0.018617+0.018586+0.018547}{3} = 0.018583 \approx 18.58\ ms$

Throughput =

Client 1 = 30904 bits / 0.018617000 s = 1.66 Mbps

Client 2 = 30904 bits / 0.018586000 s = 1.66 Mbps

Client 3 = 30904 bits / 0.018547000 s = 1.67 Mbps

Throughput avg = $\frac{1660000+1662700+1666000}{3} = 1662900\ bps \approx 1.66\ Mbps$

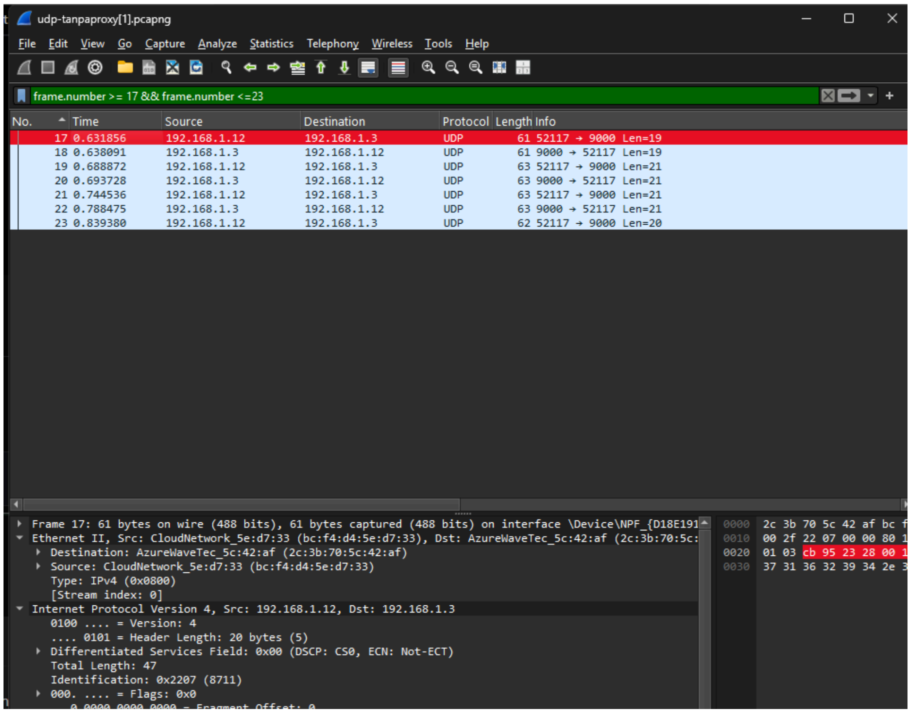
Packet Loss = 0%

Jitter = $\frac{0.000031 + 0.000039}{2} = 0.000035\ s = 0.035\ ms$

Selisih delay 1 = $|0.018586 - 0.018617| = 0.000031$ s

Selisih delay 2 = $|0.018547 - 0.018586| = 0.000039$ s

3.1.6.3.4. udp-tanpaproxy[1].pcapng


udp-tanpaproxy[1].pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

frame.number >= 17 && frame.number <=23

```
No.   Time       Source         Destination    Protocol Length Info
   17 0.631856   192.168.1.12   192.168.1.3    UDP      61 52117 → 9000 Len=19
   18 0.638091   192.168.1.3    192.168.1.12   UDP      61 9000 → 52117 Len=19
   19 0.688872   192.168.1.12   192.168.1.3    UDP      63 52117 → 9000 Len=21
   20 0.693728   192.168.1.3    192.168.1.12   UDP      63 9000 → 52117 Len=21
   21 0.744536   192.168.1.12   192.168.1.3    UDP      63 52117 → 9000 Len=21
   22 0.788475   192.168.1.3    192.168.1.12   UDP      63 9000 → 52117 Len=21
   23 0.839380   192.168.1.12   192.168.1.3    UDP      62 52117 → 9000 Len=20
```

```
> Frame 17: 61 bytes on wire (488 bits), 61 bytes captured (488 bits) on interface \Device\NPF_{D18E191
v Ethernet II, Src: CloudNetwork_5e:d7:33 (bc:f4:d4:5e:d7:33), Dst: AzureWaveTec_5c:42:af (2c:3b:70:5c:
  > Destination: AzureWaveTec_5c:42:af (2c:3b:70:5c:42:af)
  > Source: CloudNetwork_5e:d7:33 (bc:f4:d4:5e:d7:33)
    Type: IPv4 (0x0800)
    [Stream index: 0]
v Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.3
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 47
    Identification: 0x2207 (8711)
  > 000. .... = Flags: 0x0
      0 0000 0000 0000 = Fragment Offset: 0
```

```
0000  2c 3b 70 5c 42 af bc f
0010  00 2f 22 07 00 00 80 1
0020  01 03 cb 95 23 28 00 1
0030  37 31 36 32 39 34 2e 3
```

Throughput = Total data yang dikirim*8/waktu pengiriman

Throughput = 79*8/0.207524 ≈ 3045.54 bps / 3.05 kbps

Latency = Waktu Penerimaan Paket - Waktu Pengiriman Paket

Pasangan	Waktu Kirim (Ttx)	Waktu Terima (Trx)	RTT (ms)
F17-F18	0.631856	0.638091	6.235
F19-F20	0.688872	0.693728	4.856
F21-F22	0.744536	0.788475	43.939

Latency Rata - Rata = 6.235 + 4.856 + 43.939/3 ≈ 18.343ms

Packet loss 0%

Jitter:

Delay1 (17-19): T1 = 0.057016

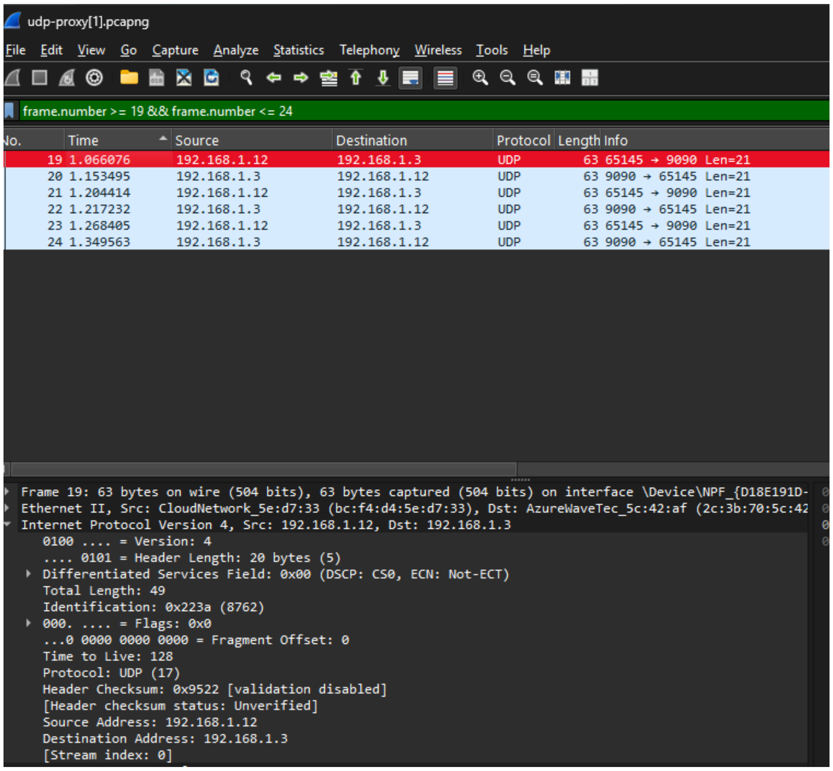
Delay2 (19-21): T2 = 0.055664

Delay3 (21-23): T3 = 0.094844

Jitter = ⅓(0.055664 - 0.057016+0.055664)

Jitter = 0.040532/3 ≈ 0.013511 detik || 13.511 ms

3.1.6.3.5. udp-proxy[1].pcapng


udp-proxy[1].pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
frame.number >= 19 && frame.number <= 24
```

```
No.   Time        Source         Destination    Protocol Length Info
   19 1.066076    192.168.1.12   192.168.1.3    UDP      63 65145 → 9090 Len=21
   20 1.153495    192.168.1.3    192.168.1.12   UDP      63 9090 → 65145 Len=21
   21 1.204414    192.168.1.12   192.168.1.3    UDP      63 65145 → 9090 Len=21
   22 1.217232    192.168.1.3    192.168.1.12   UDP      63 9090 → 65145 Len=21
   23 1.268405    192.168.1.12   192.168.1.3    UDP      63 65145 → 9090 Len=21
   24 1.349563    192.168.1.3    192.168.1.12   UDP      63 9090 → 65145 Len=21
```

```
Frame 19: 63 bytes on wire (504 bits), 63 bytes captured (504 bits) on interface \Device\NPF_{D18E191D-
Ethernet II, Src: CloudNetwork_5e:d7:33 (bc:f4:d4:5e:d7:33), Dst: AzureWaveTec_5c:42:af (2c:3b:70:5c:42
Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.3
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 49
    Identification: 0x223a (8762)
    000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: UDP (17)
    Header Checksum: 0x9522 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 192.168.1.12
    Destination Address: 192.168.1.3
    [Stream index: 0]
```

Throughput = Total Data Dikirim *8/ Waktu Pengiriman

Payload total = 21 + 21 +21 = 63 bytes

Waktu = T24 -T19 = 1.349563 - 1.066076 = 0.283487

Throughput = 504 / 0.283487 ≈ 1777.93 || 1.78 kbps

Latency = Waktu penerimaan paket - Waktu Pengirman paket

Pasangan	**Waktu Kirim (Ttx)**	**Waktu Terima (Trx)**	**RTT (ms)**
F19-F20	1.066076	1.153495	87.419
F21-F22	1.204414	1.217232	12.818

F23-F24	1.268405	1.349563	81.158

Latency Rata-rata = 87.419 + 12.818 + 81.158 / 3 ms = 181.395 ≈ 60.465 ms

Packet Loss 0%

Delay1(F19 ke F21): T21 - T19 = 1.204414 - 1.066076 = 0.138338 detik

Delay2 (F21 ke F23): T23 - T21 = 1.268405 - 1.204414 = 0.063991

Jitter : ½ (|0.063991 - 0.138338|)

Jitter : 0.074347/2 ≈ 0.0371735 detik || 37.174 ms

3.2. Diskusi hasil dan kendala pengujian

Sistem client, proxy server, dan web server secara keseluruhan mampu berkomunikasi dengan baik di semua skenario yang diuji, baik menggunakan protokol TCP maupun UDP.

Di HTTP single, client berhasil mengirim permintaan HTTP GET dan menerima respons dari web server, baik secara langsung maupun melalui proxy, dengan ukuran data yang konsisten.

Selanjutnya, pada pengujian HTTP Multi, penggunaan multithreading di client terbukti efektif untuk menghasilkan beberapa permintaan HTTP secara paralel.

Pada browser mode, client tidak hanya menerima respons HTTP, tetapi juga berhasil menyimpan konten HTML ke file lokal dan menampilkannya melalui browser.

Terakhir, pada pengujian UDP Direct dan UDP via Proxy, sistem berhasil mengirim dan menerima paket UDP secara berurutan tanpa kehilangan paket.

Selama proses pengujian, kendala utama yang kami hadapi adalah kompleksitas analisis paket di Wireshark, terutama pada pengujian multi-client. Dengan banyaknya paket TCP yang muncul akibat koneksi paralel, retransmission, dan mekanisme ACK, proses pemilihan frame untuk analisis QoS menjadi lebih rumit dan memerlukan ketelitian ekstra.

4. Kesimpulan dan saran

4.1. Kesimpulan

Berdasarkan pengujian yang telah dilakukan, sistem client, proxy server, dan web server berhasil berkomunikasi dengan baik menggunakan protokol TCP dan UDP. Proxy server berfungsi efektif sebagai perantara, khususnya pada pengujian HTTP dengan mekanisme cache yang mampu mengurangi beban web server. Analisis QoS menunjukkan tidak adanya packet loss pada seluruh pengujian, dengan latency dan throughput yang masih dalam batas wajar pada jaringan lokal. Pada pengujian UDP, penggunaan proxy menyebabkan peningkatan delay dan jitter dibandingkan komunikasi langsung karena adanya proses forwarding tambahan.

4.2. Saran

Untuk pengembangan selanjutnya, sistem proxy dapat ditingkatkan dengan pengelolaan cache yang lebih optimal. Selain itu, pengujian QoS sebaiknya dilakukan pada kondisi jaringan yang lebih bervariasi agar hasil analisis lebih representatif terhadap kondisi jaringan sebenarnya.

Referensi

https://drive.google.com/file/d/1nKQnaA649wWsWqA3IgeLOsa6yTaJ5stb/view