Perform each of the steps, and explain what is going on in your lab journal in detail. Carefully watch out for differences like TCP connections closing (or not closing).

Note that you are given code fragments. Putting the fragments together into working programs is part of the exercise!

1 HTTP/1.0, GET

$ telnet grothoff.org 80
GET / HTTP/1.0

2 HTTP/1.0 HEAD

$ telnet grothoff.org 80
HEAD / HTTP/1.0

• What happens if you use “HTTP/1.1” instead of “HTTP/1.0”?

3 HTTP/1.1

$ telnet grothoff.org 80
GET / HTTP/1.1
Host: grothoff.org

4 HTTP/1.1, Connection: close

GET / HTTP/1.1
Host: grothoff.org
Connection: close
5 HTTP/1.0, Connection: Keep-alive

GET / HTTP/1.0
Connection: Keep-alive

6 Install GNU libmicrohttpd

   libmicrohttpd-0.9.55.tar.gz
$ tar xvf libmicrohttpd-0.9.55.tar.gz
$ cd libmicrohttpd-0.9.55
$ ./configure --prefix=$HOME
$ make install

7 Start MHD HTTPD

$ cd doc/examples/
$ gcc -I$HOME/include -L$HOME/lib
   hellobrowser.c -lmicrohttpd -o hellobrowser
$ export LD_LIBRARY_PATH=$HOME/lib
$ ./hellobrowser # in another shell

7.1 Understanding hellobrowser.c

#include <microhttpd.h>

int main ()
{
   struct MHD_Daemon *daemon =
      = MHD_start_daemon (MHD_USE_AUTO | 
         MHD_USE_INTERNAL_POLLING_THREAD, 8888,
         NULL, NULL,
         &answer_to_connection, NULL,
         MHD_OPTION_END);

   if (NULL == daemon)
      return 1;
   (void) getchar ();
   MHD_stop_daemon (daemon);
   return 0;
}
static int answer_to_connection (void *cls,  
  struct MHD_Connection *connection,  
  const char *url, const char *method,  
  const char *version,  
  const char *upload_data, size_t *upload_data_size,  
  void **con_cls)
{
  const char *page = "<html><body>Hello, _browser!</body></html>";
  int ret;
  struct MHD_Response *response = MHD_create_response_from_buffer(strlen(page),  
    (void *)page, MHD_RESPMEM_PERSISTENT);
  ret = MHD_queue_response(connection, MHD_HTTP_OK, response);
  MHD_destroy_response(response);
  return ret;
}

8 Setting Response Headers

response = MHD_create_response (...);
MHD_add_response_header(response,  
  MHD_HTTP_HEADER_CONTENT_TYPE,  
  "text/html");
ret = MHD_queue_response(connection,  
  MHD_HTTP_OK,  
  response);
MHD_destroy_response(response);

Enhance your code and test it with telnet! Which headers does the response include?

9 sendfile()

int fd;
struct stat sbuf;

if (0 != strcmp(method, "GET")) return MHD_NO;
if ( (-1 == (fd = open("picture.png", O_RDONLY))) ||  
  (0 != fstat(fd, &sbuf))) {
  if (fd != -1) close(fd);
  return report_error (connection);
}
struct MHD_Response *response =  
  MHD_create_response_from_fd_at_offset (sbuf.st_size,  
    fd, 0);
MHD_add_response_header(response,  
  "Content-Type", "image/png");
ret = MHD_queue_response (connection, MHD_HTTP_OK,  
  response);

Read up on sendfile(). What is the advantage of using sendfile()?
10 siege

apt-get install siege

    $ siege -t5S http://grothoff.org/

What do you observe?

11 Apache Benchmark (ab)

apt-get install apache2-utils

    $ ab -c 25 -t5 http://grothoff.org/

What do you observe?

12 Benchmark your own MHD-based server

- top
- time BINARY
- strace -c BINARY
- iotop (requires root)

13 Incremental replies with MHD

```c
#include <stdlib.h>

static ssize_t crc (void *cls, uint64_t pos, char *buf, size_t size_max) {
    if (0 == size_max) return 0;
    if (0 == rand() % 1024 * 1024)
        return MHD_CONTENT_READER_END_OF_STREAM;
    *buf = 'b';
    return 1;
}

struct MHD_Response *response
    = MHD_create_response_from_callback
        (MHD_SIZE_UNKNOWN,
        1024,
        &crc, NULL, NULL);
```

Using telnet:

- What happens if you use the code above with a HTTP/1.0-style request?
- What happens if you use the code above with a HTTP/1.1-style request?

Using wget:

- What is the output if you use the code above?
- What happens on the wire? Use wireshark!
14 Add **body** compression

```c
#include <zlib.h>

/**
 * Try to compress a response body. Updates @a buf and @a buf_size.
 * @param [in, out] buf pointer to body to compress
 * @param [in, out] buf_size pointer to initial size of @a buf
 * @return true if buf was compressed
 */
int body_compress (void **buf, size_t *buf_size) {
    uLongf cbuf_size = compressBound (*buf_size);
    Bytef *cbuf = malloc (cbuf_size);
    int ret = compress (cbuf, &cbuf_size,
        (const Bytef *) *buf, *buf_size);
    if ((Z_OK != ret) || (cbuf_size >= *buf_size)) {
        free (cbuf); return false; }
    free (*buf);
    *buf = (void *) cbuf;
    *buf_size = (size_t) cbuf_size;
    return true;
}
```

- Add support for compression to your MHD server.
- Make sure to check the client supports compression.
- You need to link against libz

15 **Future Work (aka Bachelor’s thesis topics!)**

- Improve usability of MHD API
- Improve code coverage of tests via better test harness
Copyright (C) 2012-2018 Christian Grothoff

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free
Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with
no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.