GNU libmicrohttpd

Christian Grothoff

9.5.2013

Demo

Try me here...

`http://mhd/`

... sadly not with 10 GBit/s due to hardware unavailability.

Motivations

- HTTP(S) is the new IP — tunnel traffic over HTTP
- HTTP + HTML are a fast and portable way to create a GUI
- Full-blown HTTP Servers (Apache, etc.) are often overkill
- Extending Apache only natural with P-languages, not so much C

GNU libmicrohttpd

- Free software
- Fully HTTP/1.0 and HTTP/1.1 compliant
- Supports all common HTTP features
- Just HTTP(S) server, small footprint
- Makes limited assumptions about event handling:
  - External select/poll loop
  - Internal select/poll loop
  - One thread per connection
  - Thread pool

Response Generation

- Static buffer in memory
- Data stream (known or unknown size)
- Data stream with long polling
- From file at offset
- From file with `sendfile`
- With custom HTTP headers — and footers

Security

- Optional support for HTTPS, full X.509 support
- HTTP basic and digest authentication
- Access to client certificates
- Ability to selectively `bind` sockets
- Limiting # connections (overall, per IP), custom timeouts
- Limit memory consumption per connection

Performance

- No busy waiting, ever
- Zero copy, wherever possible
- Stream processing (GET, POST, PUT)
- Minimize `malloc`, handle **all** errors
- No re-inventing `strchr`, `strcmp`, etc.
- Clean C code, no code duplication

GNU libmicrohttpd

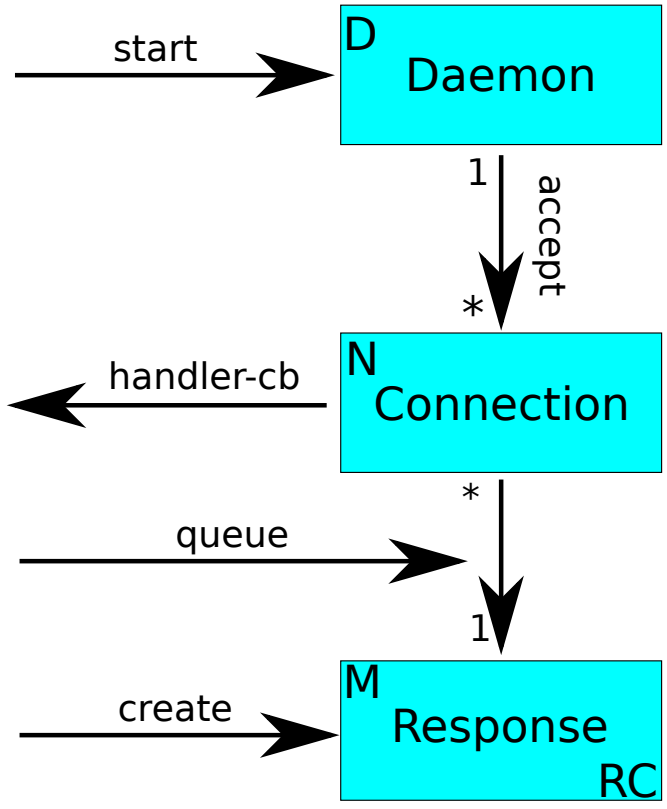
- GNU LGPL or GPL + eCoS
- Library binary can be as small as 32k
- We reportedly have users on systems with 50 Mhz processors with HTTPS
- We have users working with MHD on systems with 64 kb RAM

"I also ran oprofile on the system while streaming about 7gbps to (simulated) ipads and while ramping up 1000s of streams (which causes high rate of HTTP requests to read the Apple HLS playlists). libmicrohttpd barely registers as cpu usage." – MHD user

Applications using MHD

- GNUnet, P4P Portal
- Gnome Music Player Client, Kiwix, XMBC, OpenVAS
- Psensor, Disk Nukem, Flat8, Fawkes, Conky, CallHome
- OpenDIAS, Techne, Cables communication project
- Open Lightning Architecture, OpenZWave, libhttpserver
- Plus many non-free applications (such as TVs, surveilance cameras, network appliances, etc.)

The MHD API


start
D
Daemon
1
accept
*
handler-cb
N
Connection
*
queue
1
create
M
Response
RC

The MHD API

- MHD_start_daemon
- MHD_stop_daemon
- MHD_create_response_from_...
- MHD_queue_response
- MHD_destroy_response
- +30 other functions you might not need

Hello Browser!

```
#include <microhttpd.h>

int main ()
{
  struct MHD_Daemon *daemon;
  daemon = MHD_start_daemon (MHD_USE_SELECT_INTERNALLY, 1080, NULL, NULL,
                             &answer_to_connection, NULL, MHD_OPTION_END);
  if (NULL == daemon)
    return 1;
  getchar ();
  MHD_stop_daemon (daemon);
  return 0;
}
```

Hello Browser: Responding

```
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>";
  struct MHD_Response *response;
  int ret;

  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;
}
```

Proper GET Handling

```
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) {
  struct MHD_Response *response;
  int fd, ret;
  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);
  }
  response =
    MHD_create_response_from_fd_at_offset (sbuf.st_size, fd, 0);
  MHD_add_response_header (response, "Content-Type", "image/jpg");
  ret = MHD_queue_response (connection, MHD_HTTP_OK, response);
  MHD_destroy_response (response);
  return ret;
}
```

... and the error reporting

```
static int
report_error (struct MHD_Connection *connection)
{
  const char *errorstr =
   "<html><body>An internal server error has occured!</body></html>";
  struct MHD_Response *response;
  int ret;

  response =
    MHD_create_response_from_buffer (strlen (errorstr),
             (void *) errorstr,
             MHD_RESPMEM_PERSISTENT);
  if (NULL != response)
  {
    ret =
       MHD_queue_response (connection, MHD_HTTP_INTERNAL_SERVER_ERROR,
                           response);
     MHD_destroy_response (response);
     return ret;
  }
  return MHD_NO;
}
```

Demo

Let's look at the code...

Documentation

- Reference Manual
- Tutorial
- Doxygen source code comments
- Examples
- Testcases

Future Work

- Add support for WebSockets
- Add support for SPDY/HTTP 2.0
- Improve code coverage of tests

RTFL