分散システム特論レポート

Hypertext Transfer Protocol – HTTP/1.1

石津 健太郎

2002 年 2 月

1 HTTP の通信

HTTP は，分散した複数の種類の情報を統合するアプリケーション層のプロトコルである．この通信は，クライアントの要求とサーバの応答から成り立っている．

通信方式は RFC822 で規定する以下の書式である．

HTTP-message = Request — Response

以下に，HTTP 通信の例を示す．

- 要求 (Request)

```
GET /sample.txt HTTP/1.1
Host: cat.csce.kyushu-u.ac.jp
Range: bytes=50-100
```

- 応答 (Response)

```
HTTP/1.1 200 Ok
Date: Thu, 08 Nov 2001 09:29:52 GMT
Server: Apache/1.3.19 (Unix)  (Vine/Linux)
Last-Modified: Tue, 06 Nov 2001 11:40:17 GMT
ETag: "5b1be-d9-3be7cc21"
Accept-Ranges: bytes
Content-Length: 33
Content-Type: text/plain

Hello!
This is a test.
Bye.
```

2 要求 (Request) と応答 (Response)

HTTP の要求と応答について説明する．

2.1 要求 (Request)

```
Request =
          Request-Line
          *(( general-header | request-header | entity-header ) CRLF)
          CRLF
          [ message-body ]

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Method = "OPTIONS" | "GET" | "HEAD" | "POST" | "PUT" | "DELETE"
       | "TRACE"| "CONNECT" | extension-method

Request-URI = "*" | absoluteURI | abs_path | authority

HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT
```

要求は，上に示すように Request-Line と 0 個以上のヘッダ行，それから空行を挟んで message-body から成っている．

Request-Line は，メソッド名，Request-URI, HTTP のバージョンと改行コードから成っている．ここで，メソッドとは要求の種類を示す文字列であり，上に示すようなものが定義されている．

HTTP-Version は，”HTTP/” という文字列の後に 1 桁の数字 2 つを”.”で挟んだものである．

このようにして定義した HTTP の要求の例を以下に示す．

- 例 1

```
GET http://dontaku.csce.kyushu-u.ac.jp/members2001.html HTTP/1.0
```

- 例 2

```
HEAD /graphics/hiyami.gif HTTP/1.1
Host: cat.csce.kyushu-u.ac.jp
```

- 例 3

```
OPTIONS * HTTP/1.1
Host: cat.csce.kyushu-u.ac.jp
```

例で示したヘッダの詳細については後述する．

2.2 応答 (Response)

```
Response = Status-Line
           *(( general-header | response-header | entity-header   ) CRLF)
           CRLF
           [ message-body ]
```

```
Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

Status-Code =
          "100"  ; Continue
        | "101"  ; Switching Protocols
        | "200"  ; OK
        | "201"  ; Created
        | "300"  ; Multiple Choices
        | "301"  ; Moved Permanently
        | "400"  ; Bad Request
        | "401"  ; Unauthorized
        | "500"  ; Internal Server Error
        | "501"  ; Not Implemented
        | .. ..
```

HTTP の応答は，Status-Line，0 個以上のヘッダ，それから空行を挟んで message-body が続く．
Status-Line は，HTTP-Version，Status-Code，Reason-Phrase をスペースで連結したものである．

Status-Line の例を以下に示す．

- 例 1

```
HTTP/1.1 200 Ok
```

- 例 2

```
HTTP/1.0 501 Not Implimented
```

3 ヘッダ

ヘッダは，種類ごとに大きく分けて general-header, request-header, response-header, entity-header の 4 種が以下のように定義されている．

```
general-header
        = Cache-Control | Connection | Date
          | Transfer-Encoding | Upgrade | Via | ・  ・  ・

request-header
        = Accept | Accept-Charset | Authorization
          | Host | If-Modified-Since | If-Range
          | Range | User-Agent | ・  ・  ・

response-header
```

```
        = Accept-Ranges | Age | ETag
          | Location | Retry-After | ・  ・  ・

entity-header
        =  Content-Language | Content-Length | Content-MD5
          | Content-Type | Expires | Last-Modified | ・  ・  ・
```

3.1 Date

```
Sun, 06 Nov 1994 08:49:37 GMT        ; RFC 822, updated by RFC 1123
Sunday, 06-Nov-94 08:49:37 GMT       ; RFC 850, obsoleted by RFC 1036
Sun Nov  6 08:49:37 1994             ; ANSI C's asctime() format
```

3.2 Content-Type

```
Content-Type = "Content-Type"  ":"  media-type
media-type = type "/" subtype *( ";" parameter )

ex. Content-Type: text/html; charset=ISO-8859-4
```

3.3 Etag

リソースに固有のタグで，キャッシュ制御に利用する．

3.4 If-Match

ヘッダの Etag と実体の Etag が一致する場合、要求されたメソッドを処理

```
If-Match: "xyzzy", "r2d2xxxx", "c3piozzzz"
```

3.5 If-None-Match

3.6 If-Unmodified-Since

指定した日時から変更されていない場合には，サーバは要求された操作を行わない．

```
If-Unmodified-Since: Sat, 29 Oct 1994 19:43:31 GMT
```

3.7 If-Modified-Since

3.8 From

利用者のメールアドレスを示す．サーバへ Request を送るときに一緒に送る．

```
From: ishidu@hogehoge.org
```

3.9 Referer

どこから URI を得たか

```
Referer: http://www.kyushu-u.ac.jp/index.html
```

3.10 Range

リソースを部分的に取得する場合に，範囲を指定．

```
Range: bytes=0-499
Range: bytes=500-999
Range: bytes=-500
Range: bytes=9500-
```

3.11 Retry-After

指定した時間に，サーバがクライアントに再度要求を行なわせる．

```
Retry-After: Fri, 31 Dec 2001 23:59:59 GMT    ; 指定した時刻
Retry-After: 120                              ;120 秒後
```

3.12 Connection

接続を切断する．

```
Connection: close
```

3.13 Location

URI の指定．

```
Location: http://www.w3.org/pub/WWW/People.html
```

3.14 Upgrade

サーバがクライアントに対して，新しい通信プロトコルを通知する．Status Code 101 (Switching Protocols) と共に使用する．

```
Upgrade: HTTP/2.0, SHTTP/1.3, IRC/6.9, RTA/x11
```

4 メソッド (Method)

サービスの要求ヘッダにおいて，サービスの種類を指定する．

```
Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Method  = "OPTIONS" | "GET" | "HEAD" | "POST" | "PUT"
          | "DELETE" | "TRACE"| "CONNECT" | extension-method
```

4.1 HEAD

URI で示す実体のヘッダだけを要求 (BODY は転送しない)

```
HEAD / HTTP/1.1
Host: cat.csce.kyushu-u.ac.jp

HTTP/1.1 200 OK
Date: Thu, 08 Nov 2001 10:52:46 GMT
Server: Apache/1.3.19 (Unix)  (Vine/Linux)
Last-Modified: Tue, 06 Nov 2001 11:40:17 GMT
ETag: "5b1be-d9-3be7cc21"
Accept-Ranges: bytes
Content-Length: 217
Content-Type: text/html
```

4.2 POST

message-body として送るメッセージをサーバに渡す. CGI に広く使用されるが , POST メソッドによる動作はサーバに依存している .

```
POST /~ishidu/test.cgi
Host: cat.csce.kyushu-u.ac.jp
Content-Length: 10

0123456789

HTTP/1.1 200 OK
Date: Thu, 08 Nov 2001 11:00:43 GMT
Server: Apache/1.3.12 (Unix) PHP/3.0.14-i18n-ja
Transfer-Encoding: chunked
Content-Type: text/html
```

4.3 OPTIONS

指定する URI において使用可能な通信方法を要求

```
OPTIONS * HTTP/1.1
Host: cat.csce.kyushu-u.ac.jp
```

```
HTTP/1.1 200 OK
Date: Thu, 08 Nov 2001 10:52:10 GMT
Server: Apache/1.3.20 (Unix)
Content-Length: 0
Allow: GET, HEAD, OPTIONS, TRACE
```

5 Status Code

応答の種類を表す．

数字だけの Status-Code と 人間が解釈するための Reason-Phrase がある．Reason-Phrase は人間が読むためのものなので，厳密には定義されていない．

```
200 Ok
400 Bad Request
```

5.1 Informational 1xx

status line と optional header だけで，body はない．

5.1.1 100 Continue

サーバがクライアントからの要求の続きを求めている

5.1.2 101 Switching Protocols

HTTP をより新しいバージョンに切替えるなど

5.2 Successful 2xx

要求の受信、解析、受理が成功した場合に返す．

5.2.1 200 OK

5.2.2 201 Created

作成した資源の URI を Location: ヘッダで示す.

5.2.3 206 Partial Content

要求ヘッダで Range: を指定して部分的に取得した場合など.

5.3 Redirection 3xx

要求を他のサーバにリダイレクトする．
この status code を受けたクライアントは，メソッドが GET か HEAD の場合には，利用者の確認なしに次の要求を行える．

5.3.1 300 Multiple Choices

複数のリダイレクト先の URI を Location ヘッダで指定する．

5.3.2 301 Moved Permanently

要求された URI は永久に移動し、これ以降は存在しない．
移動先は location ヘッダで指定し，body には新しい URI へのリンクや注意書きを含める．

5.3.3 307 Temporary Redirect

要求された資源が一時的に別の URI にある

5.4 Client Error 4xx

クライアント側のエラーを示す．
アプリケーションは応答メッセージを全て利用者に表示すること．

5.4.1 400 Bad Request

HTTP の文法に従っていない．

5.4.2 401 Unauthorized

5.4.3 402 Payment Required

5.4.4 404 Not Found

5.4.5 410 Gone

5.4.6 412 Precondition Failed

クライアントから検査用のヘッダ (If-Match など) に対して、検査に失敗した場合．

5.4.7 413 Request Entity Too Large

5.5 Server Error 5xx

サーバ側のエラーを示す．

5.5.1 500 Internal Server Error

サーバが要求に対応できない不慮の事態．

5.5.2 501 Not Implemented

対応していないメソッドを要求された場合．

5.5.3 504 Gateway Timeout

5.5.4 505 HTTP Version Not Supported

6 通信の継続

要求→応答 の後で接続を切断しない．version 1.1 からデフォルトになった．
利点としては，

1. 要求のたびに再接続の必要が無い
    - TCP 接続を行う時間と帯域を節約
    - 接続に伴うサーバとルータの CPU 時間を節約
2. 通信のパイプライン化
    - 応答が戻ってくる前に複数の要求
    - 要求を送出した順に応答を受信

要求ヘッダに「Connection: close」を付けると接続を切断する．

6.1 HTTP/1.0

```
[ishidu@cat ds]$ telnet dontaku 80
Trying 133.5.22.128...
Connected to dontaku.csce.kyushu-u.ac.jp.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 10 Jan 2002 09:34:55 GMT
Server: Apache/1.3.3 (Unix) PHP/3.0.5
Connection: close
Content-Type: text/html

Connection closed by foreign host.
[ishidu@cat ds]$
```

6.2 HTTP/1.1

```
[ishidu@cat ds]$ telnet dontaku 80
Trying 133.5.22.128...
Connected to dontaku.csce.kyushu-u.ac.jp.
Escape character is '^]'.
HEAD / HTTP/1.1
Host: cat.csce.kyushu-u.ac.jp.

HTTP/1.1 200 OK
Date: Thu, 10 Jan 2002 09:37:08 GMT
Server: Apache/1.3.3 (Unix) PHP/3.0.5
Content-Type: text/html

HEAD / HTTP/1.1
Host: cat.csce.kyushu-u.ac.jp.
Connection: close

HTTP/1.1 200 OK
Date: Thu, 10 Jan 2002 09:37:24 GMT
Server: Apache/1.3.3 (Unix) PHP/3.0.5
Connection: close
Content-Type: text/html

Connection closed by foreign host.
[ishidu@cat ds]$
```

7 キャッシュ

キャッシュの期限切れを検出して要求の送信回数を減らすか，応答の受け取りが不必要な場合を判断して資源の受けとりをしない．

キャッシュを使用する条件 (以下のいずれかが当てはまるとき)

- 要求されたコンテンツが，キャッシュと同じであるという応答をサーバから得る
- 十分に新しいということが既知である
- 304(Not Modified), 305(Proxy Redirect), エラー (4xx or 5xx) の応答を得る

7.1 キャッシュの新しさの判断

1. 期限切れ指定モデル
   - サーバによる期限切れの指定
   - キャッシュの作成時間による期限切れの決定

- 各サーバでキャッシュされた合計時間 + ネットワーク転送に要した時間

2. 有効性問い合わせモデル
   - Last-Modified を比較
   - Etag を比較

7.2 ヘッダ

キャッシュ制御に関わるヘッダ

```
cache-directive = cache-request-directive | cache-response-directive

cache-request-directive = "no-cache"
                          | "no-store"
                          | "max-age" "=" delta-seconds
                          | "max-stale" [ "=" delta-seconds ]
                          | "min-fresh" "=" delta-seconds
                          | "no-transform"
                          | "only-if-cached"
                          | cache-extension
```

8 実装

Apache 1.3.22

8.1 Etag 作成

Etag を作成する関数

```
API_EXPORT(char *) ap_make_etag(    ・    ・    ・    )
{
            ・    ・    ・
    /* ファイルの i-node, サイズ、更新日から Etag を計算 */
    etag = ap_psprintf(r->pool,
                "%s\"%lx-%lx-%lx\"", weak,
                (unsigned long) r->finfo.st_ino,
                (unsigned long) r->finfo.st_size,
                (unsigned long) r->mtime);
           ・    ・    ・
}
```

8.2 Etag を用いたキャッシュ制御応答

Etag を用いてサーバがクライアントのキャッシュ制御を行なう部分 .

```
API_EXPORT(int) ap_meets_conditions(request_rec *r)
{             ・    ・    ・
        /* If-Match がヘッダに存在し、かつ、その値が "*" ではなく、
         * Etag がこのどの値とも一致しないならば、412(Precondition Failed)
         * のステータスを返す
         */
        if ((if_match = ap_table_get(r->headers_in, "If-Match"))!= NULL) {
                  if (if_match[0] != '*' && (etag == NULL || etag[0] == 'W' ||
                                      !ap_find_list_item(r->pool, if_match, etag)) {
                           return HTTP_PRECONDITION_FAILED;
                  }
        } else {
                  /* そうでなくて、If-Unmodified-Since ヘッダがあり、要求され
る
                   * 資源がこの値により指定された日時から変更されていない場合は
、
                   * 412(Precondition Failed) を返す
                   */
                  if (if_unmodified != NULL) {
                            if ((ius != BAD_DATE) && (mtime > ius)) {
                                     return HTTP_PRECONDITION_FAILED;
                            }
                  }
        }
        /* If-None-Match ヘッダが存在し、その値が "*" であるか、
         * 実体の Etag が値のいずれかに一致する場合は、失敗。
          * GET か HEAD のとき、失敗というのは、304(Not Modified)。
          * その他のメソッドのとき、失敗というのは、412(Precondition Failed)
          */
         if_nonematch = ap_table_get(r->headers_in, "If-None-Match");
         if (if_nonematch != NULL) {
                  if (r->method_number == M_GET) {
                          if (if_nonematch[0] == '*')
                                  return HTTP_NOT_MODIFIED;
                                     ・   ・   ・   ・   ・
                  }
                  else if (if_nonematch[0] == '*' ||
                           (etag != NULL && ap_find_list_item(r->pool, if_nonema
tch, etag))) {
                          return HTTP_PRECONDITION_FAILED;
                  }
         } else if ((r->method_number == M_GET)
                    && ((if_modified_since =    ・ ・ ・ ・ ・) != NULL)) {
```

```
                    /* If-Modified-Since ヘッダが存在し、GET か HEAD のとき、
                     * ヘッダによって指定された日時から変更されていない場合は、304
(Not Modified)
                     */
                    if ((ims >= mtime) && (ims <= r->request_time)) {
                            return HTTP_NOT_MODIFIED;
                    }
            }
}
```