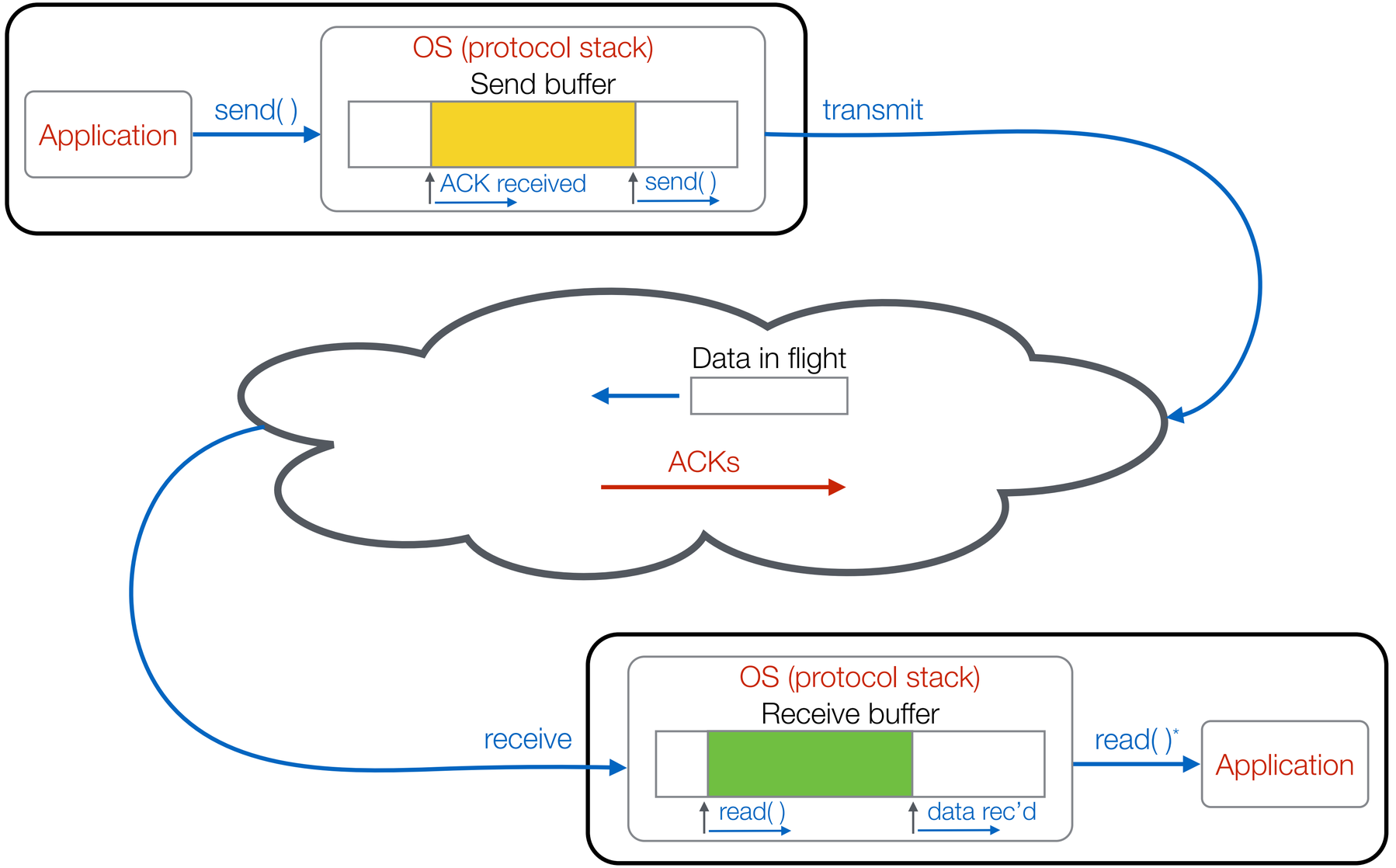
CS 725/825 & IT 725

Lecture 8

Application Layer

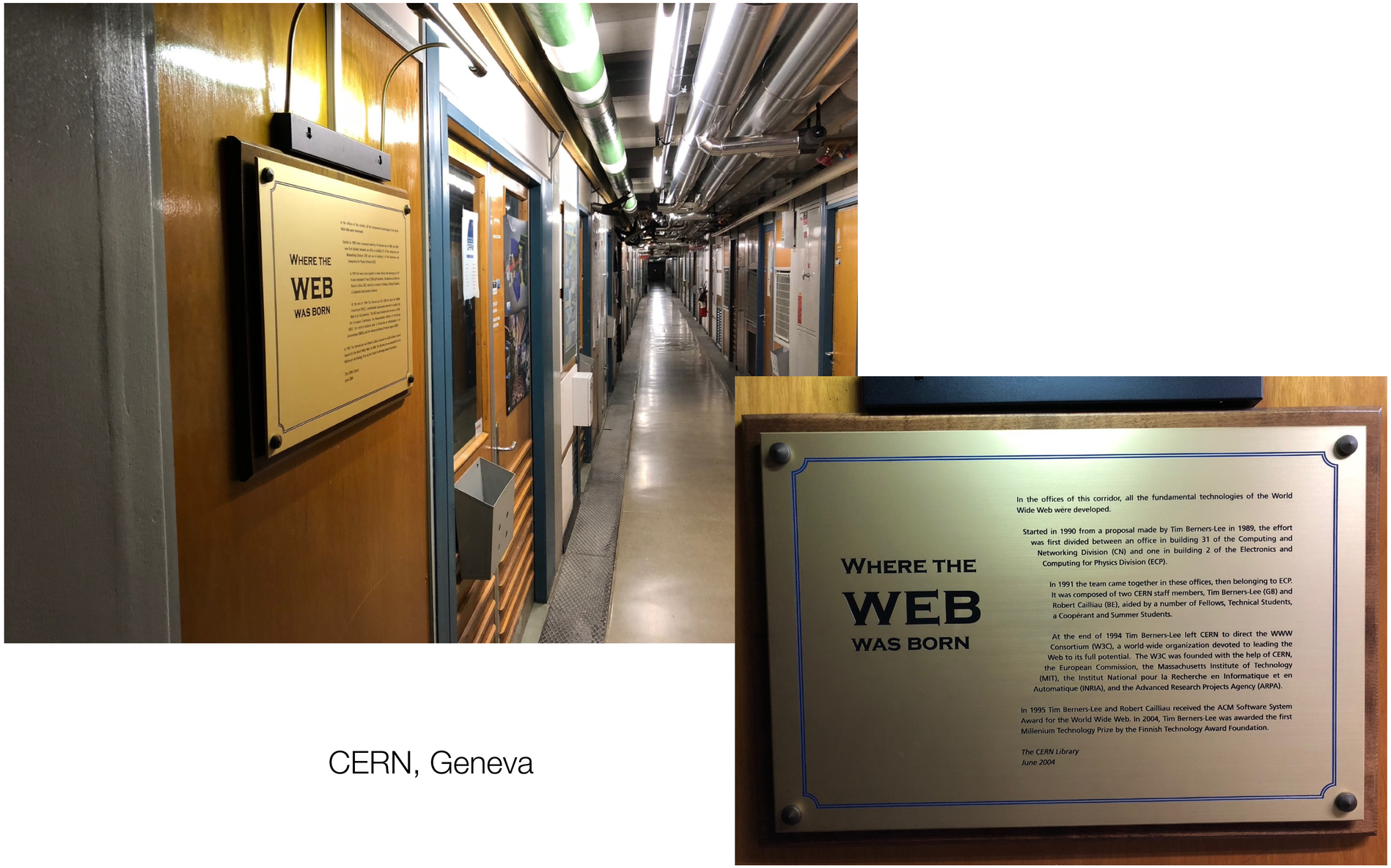
September 23, 2024

Data flow and buffering



(*) many APIs call the read() operation "receive" (eg: recv()), read is used here to avoid confusion with receiving data on an interface

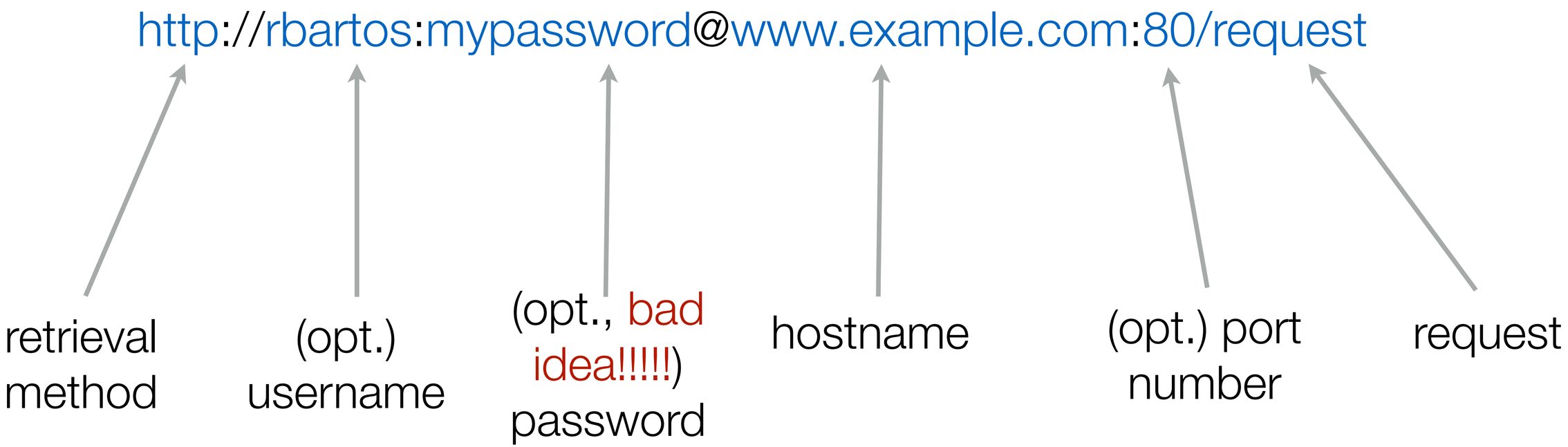
HTTP/HTML History



CERN, Geneva

HTTP

- HyperText Transfer Protocol (HTTP)
- URL (Universal Resource Locator)



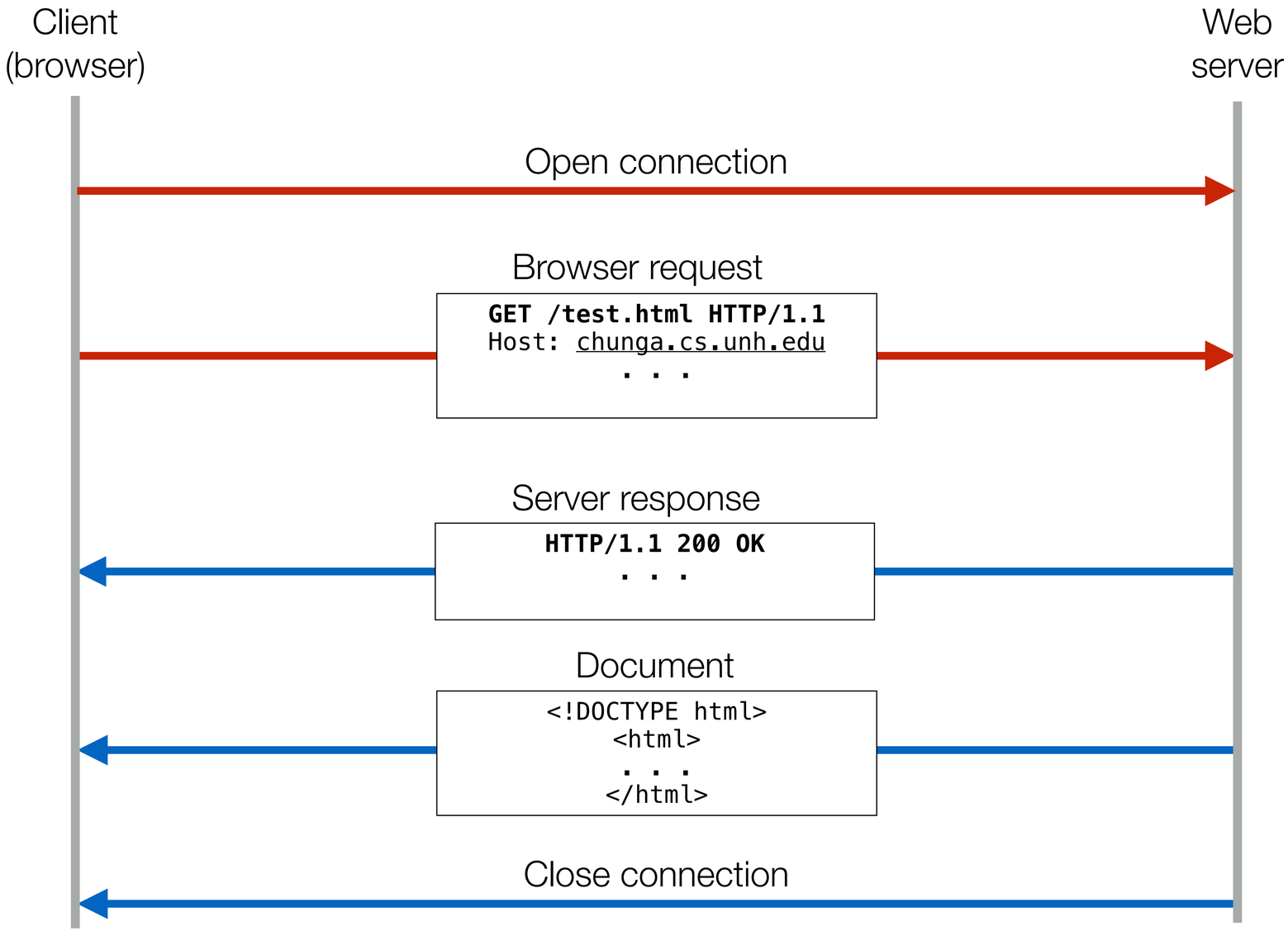
HTTP

- HyperText Transfer Protocol (HTTP)
- Runs on to of (reliable, transparent, connection oriented) TCP
- A stateless…
- … request/response protocol.
- protocol and payload not secured by default
- “work in progress”: HTTP/1.1 → HTTP/2 → HTTP/3

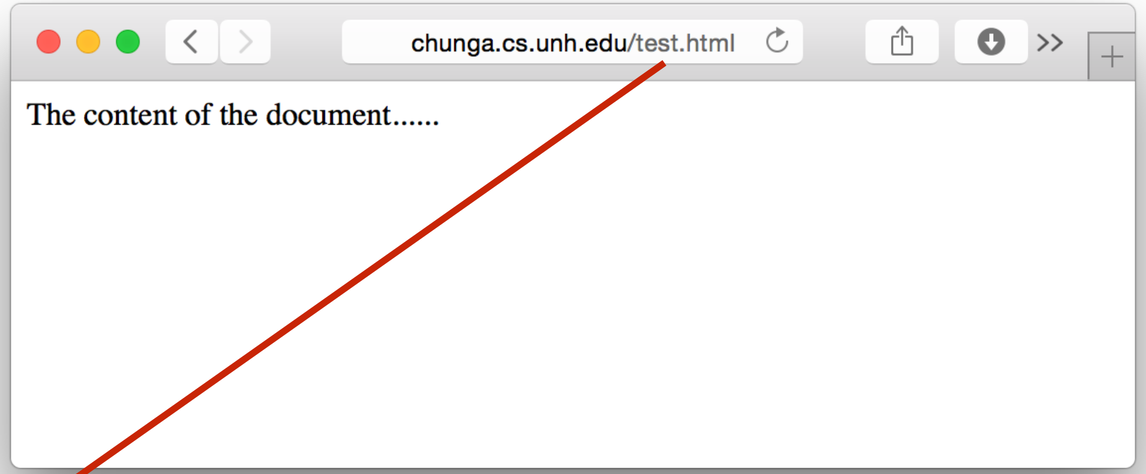
Main HTTP Methods

- GET
  - client requests data from the server
  - server sends data
- POST
  - client informs server about data to be delivered
  - client sends data
  - server sends data (response) back
- also: HEAD, PUT, DELETE, TRACE, CONNECT, …

HTTP GET Request



GET Request Details




```
GET /test.html HTTP/1.1
Host: chunga.cs.unh.edu
DNT: 1
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2)↩
            AppleWebKit/600.3.18 (KHTML, like Gecko)↩
            Version/8.0.3 Safari/600.3.18
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Cache-Control: max-age=0
```

Browser request

Terminated by a blank line

GET Response Details

```
HTTP/1.1 200 OK
Date: Wed, 25 Feb 2015 21:24:38 GMT
Server: Apache/2.4.10 (Unix) PHP/5.5.14
Last-Modified: Wed, 25 Feb 2015 21:17:51 GMT
ETag: "86-50ff02af7bdc0"
Accept-Ranges: bytes
Content-Length: 134
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<!DOCTYPE html>
<html>
<head>
    <title>Title of the document</title>
</head>
<body>
    The content of the document......
</body>
</html>
```

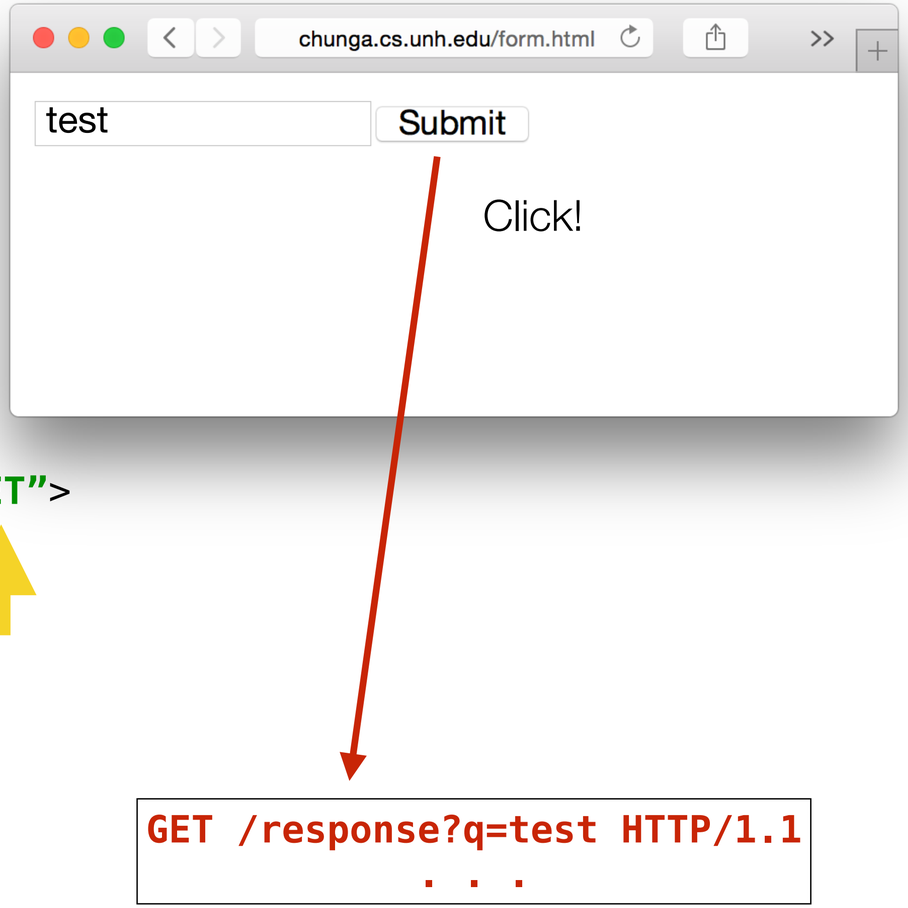
Server response

blank line

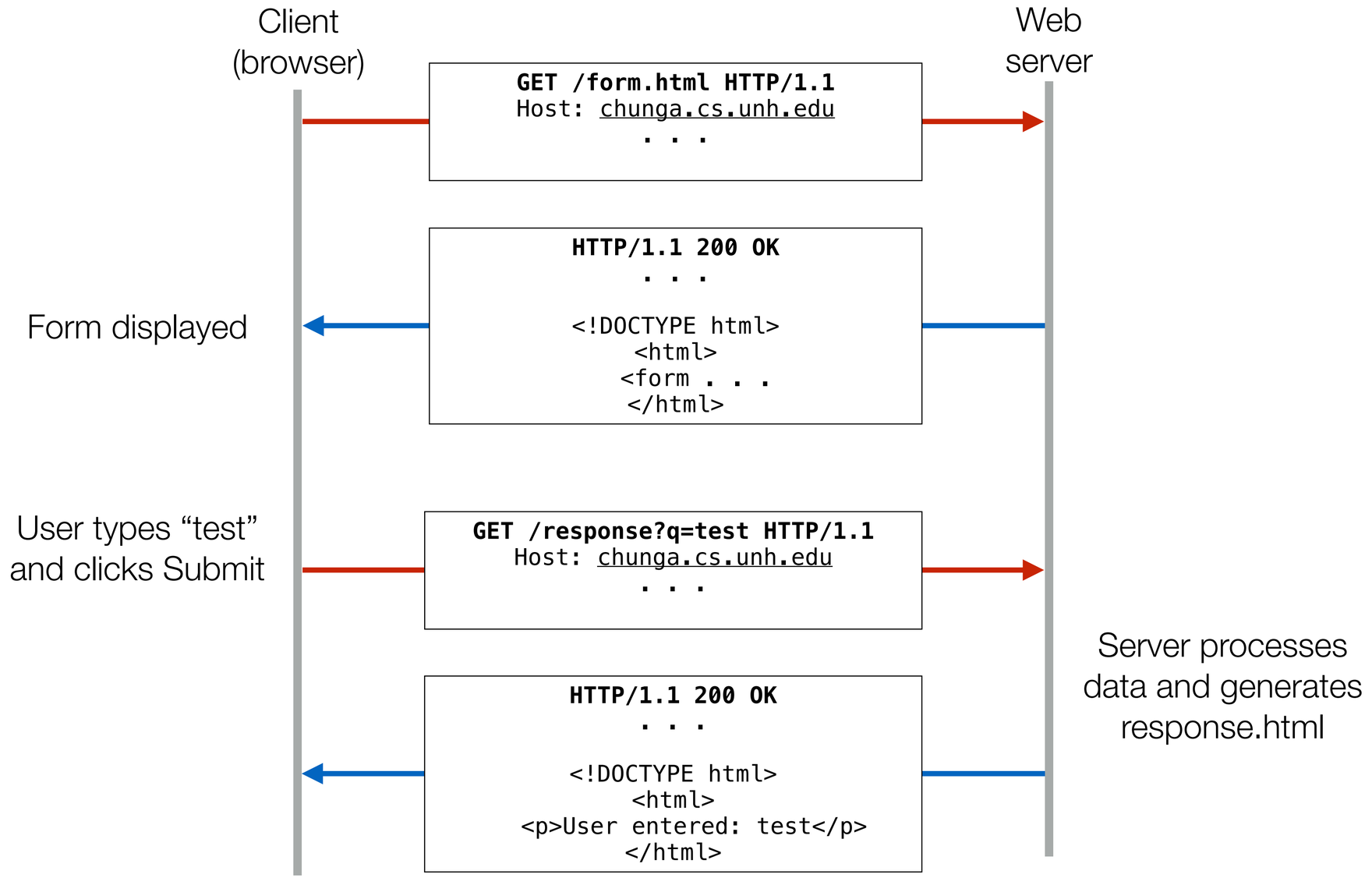
Document

Sending data: GET

```
<!DOCTYPE html>
<html>
<head>
    <title>Form</title>
</head>
<body>
    <form action="/response" method="GET">
        <input type="text" name="q">
        <input type="submit">
    </form>
</body>
</html>
```



Sending date: GET



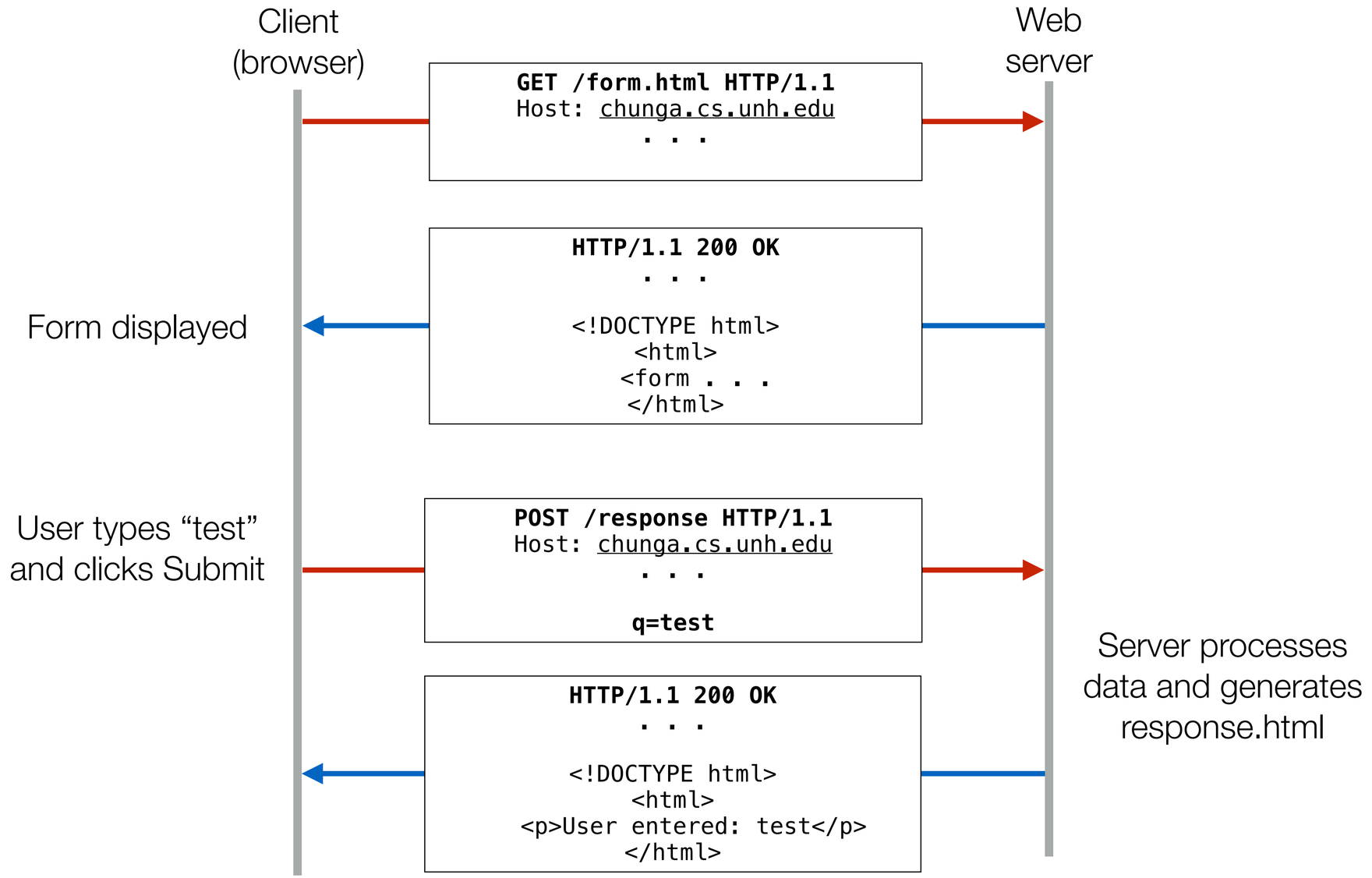
Sending data: POST

```
<!DOCTYPE html>
<html>
<head>
    <title>Form</title>
</head>
<body>
    <form action="/response" method="POST">
        <input type="text" name="q">
        <input type="submit">
    </form>
</body>
</html>
```


```
POST /response HTTP/1.1
Host: chunga.cs.unh.edu
Content-Type: application/x-www-form-urlencoded
. . .
Referer: http://chunga.cs.unh.edu/form.html
. . .

q=test
```

Sending data: POST



Not just forms: AJAX

```
<script>
  $("form").submit(function(f) {
    f.preventDefault();
    $.ajax({
      url: "/request",
      data: “q=“+$("#q").val(),
      type: "GET",
      success: function(data) {
        var results = JSON.parse(data);
        $("#result").html(results["result"]);
      },
      error: function(e) {
        alert("Error: "+e.status+": "+e.statusText);
      }
    });
  });
</script>
```

GET /request?q=test HTTP/1.1

{"result":"server response"}

HTTP transaction in general