HyperText Transfer Protocol

The HyperText Transfer Protocol (HTTP) is used to define how the client-server programs can be written to retrieve web pages from the Web.

An HTTP client sends a request; an HTTP server returns a response.

The server uses the port number 80; the client uses a temporary port number.

HTTP uses the services of TCP, and connection-oriented and reliable protocol.

Nonpersistent versus Persistent Connections

- If the web pages are located on different servers, we must create a new TCP connection for retrieving each object.

- If the objects are located on the same server, we have two choices:

1. *nonpersistent connection:* retrieve each object using a new TCP connection.
2. *persistent connection*: make a TCP connection and retrieve them all.

Figure shows an example of a nonpersistent connection. The client needs to access a file that contains one link to an image. The text file and image are located on the same server.

Here we need two connections. For each connection, TCP requires at least three handshake messages to establish the connection, but the request can be sent with the third one. After the connection is established, the object can be transferred. After receiving an object, another three handshake messages are needed to terminate the connection.

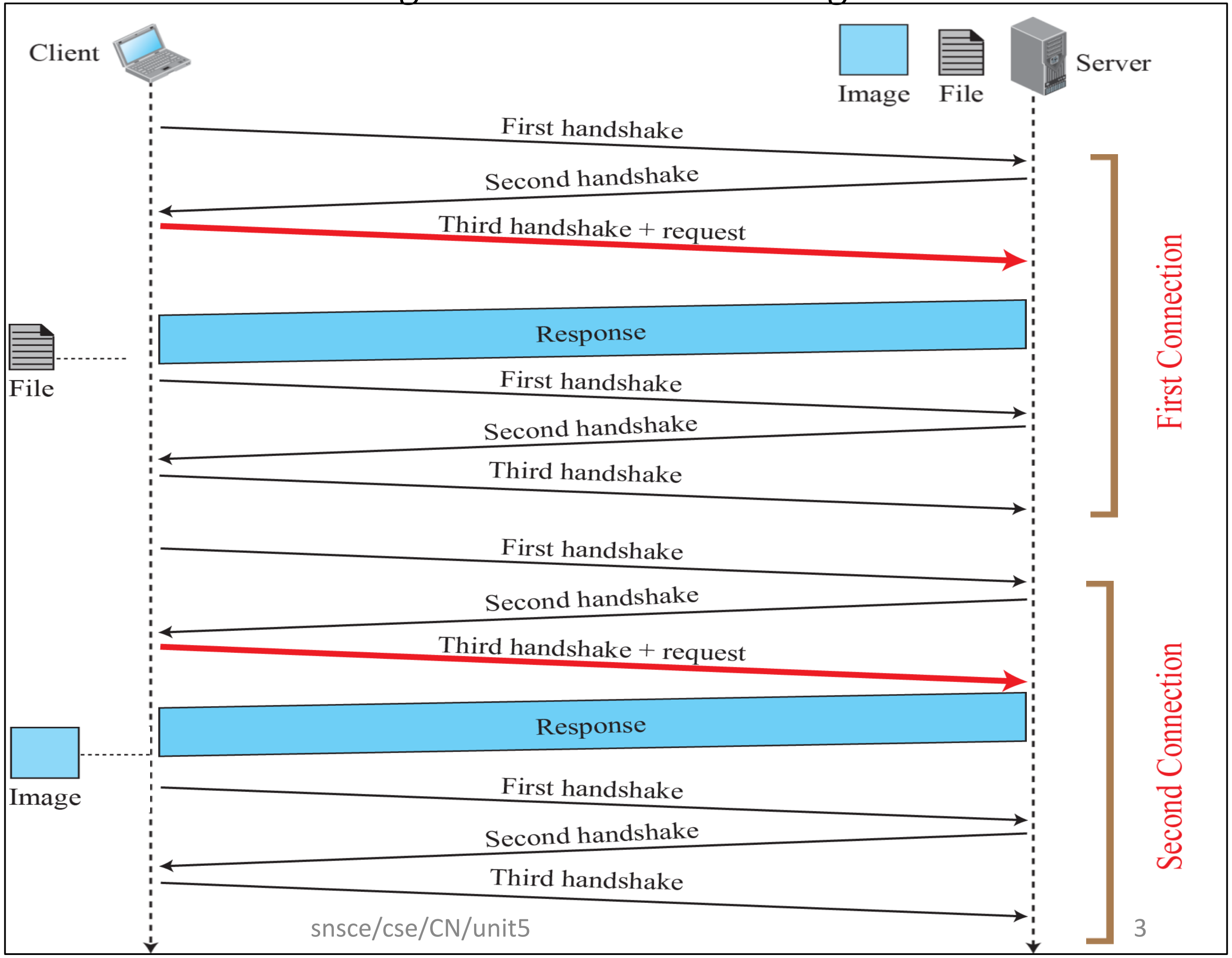
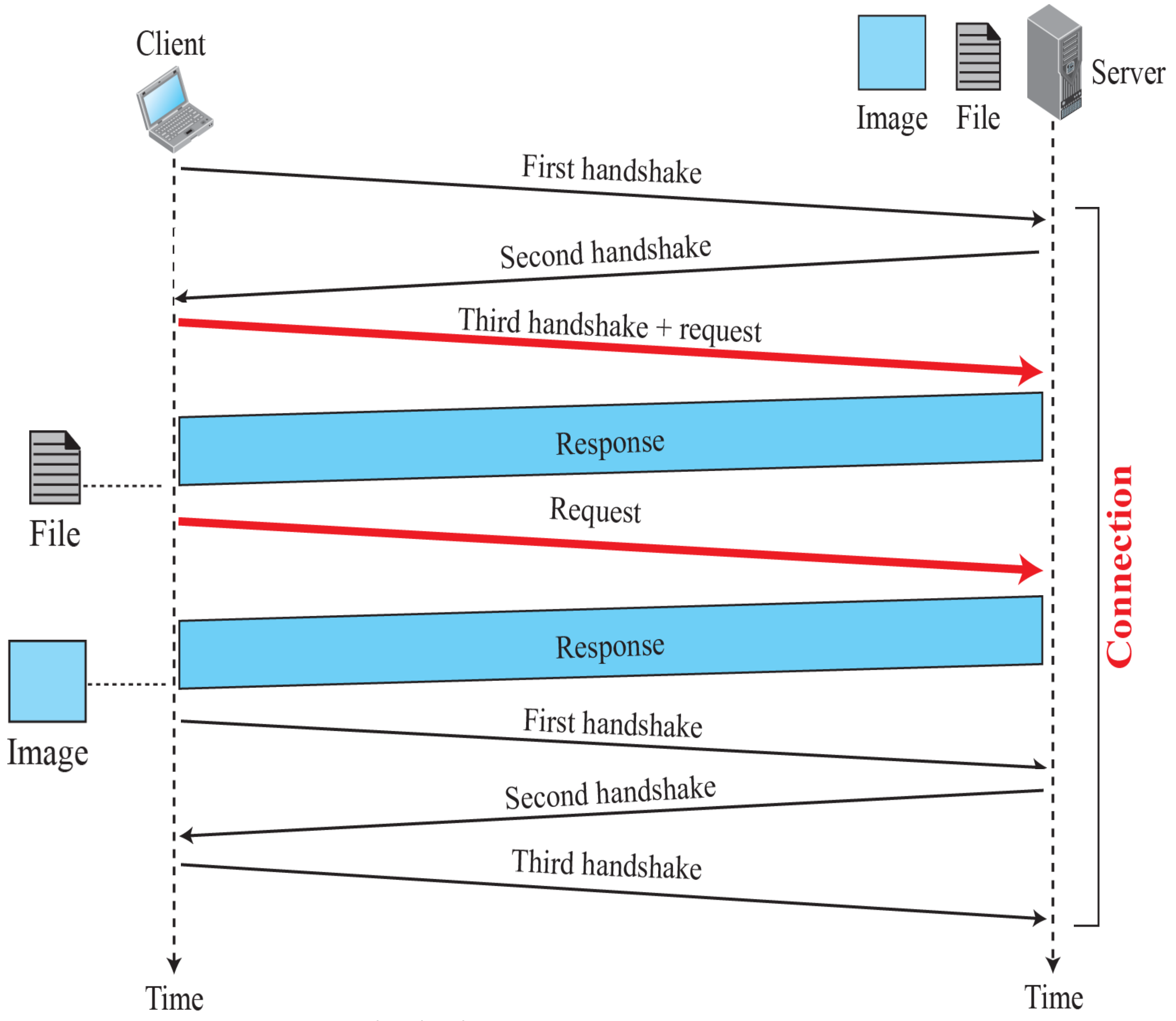
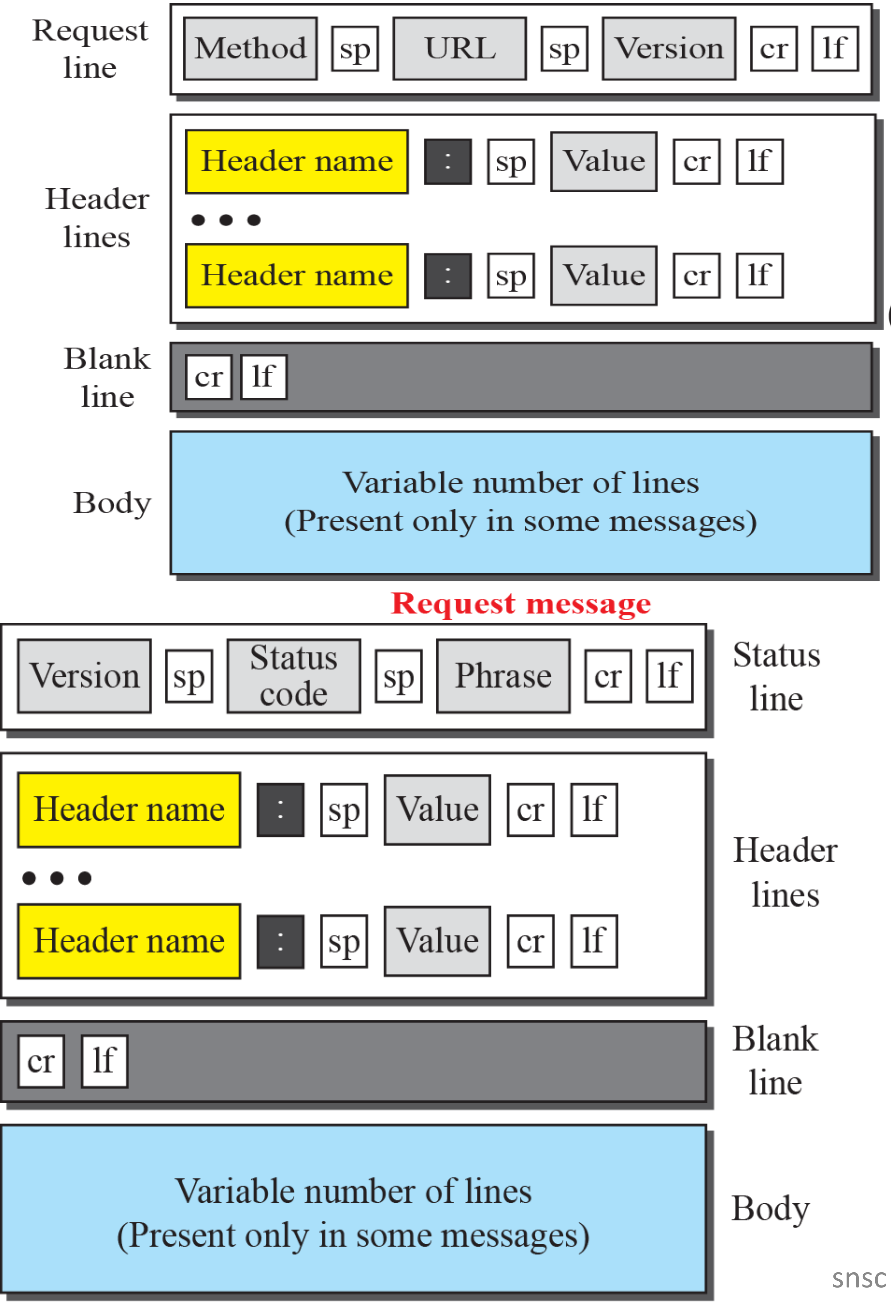


Figure shows the same scenario using a persistent connection. Only one connection establishment and connection termination is used, but the request for the image is sent separately.



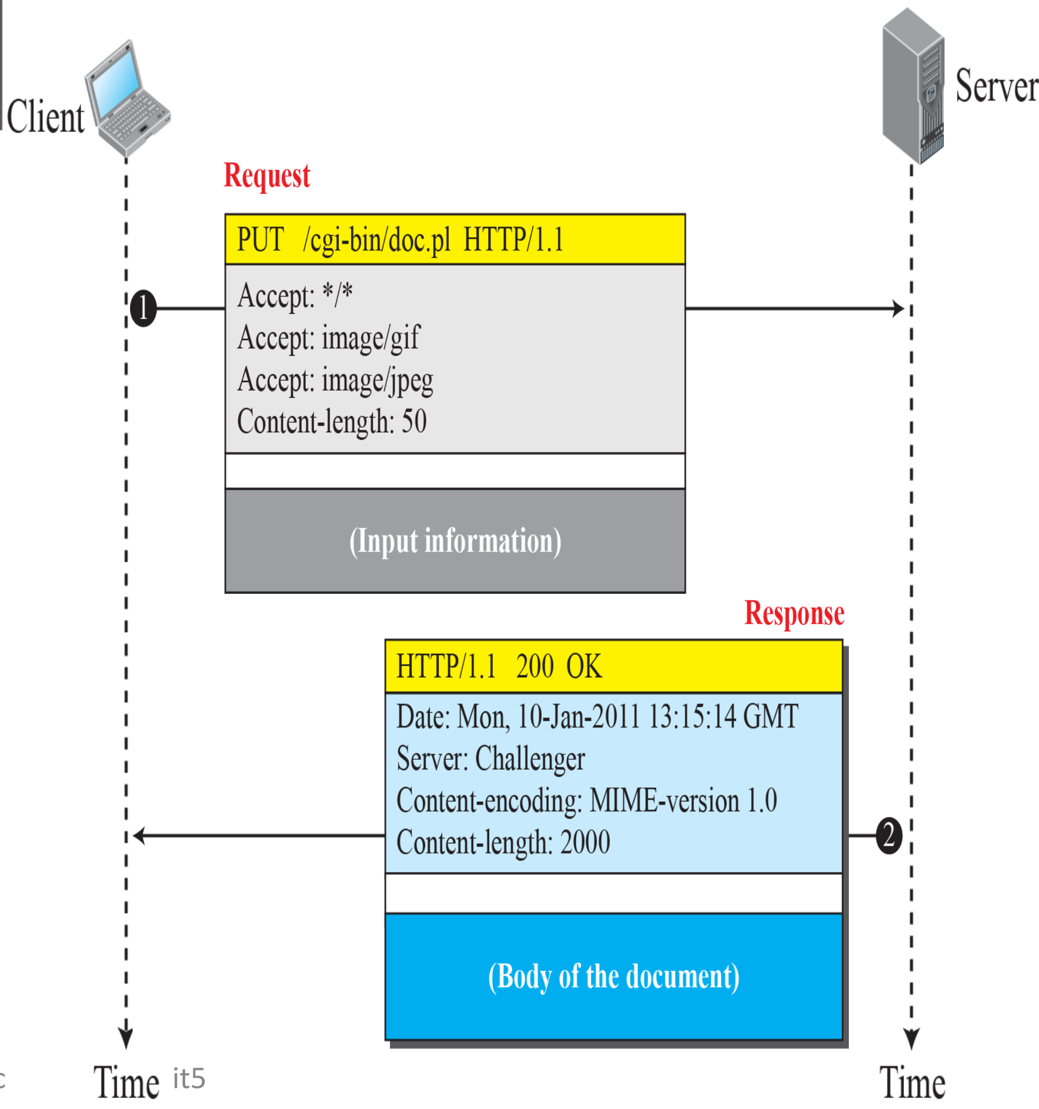
rmats of the request and response messages

Legend sp: Space cr: Carriage Return lf: Line Feed



Request message

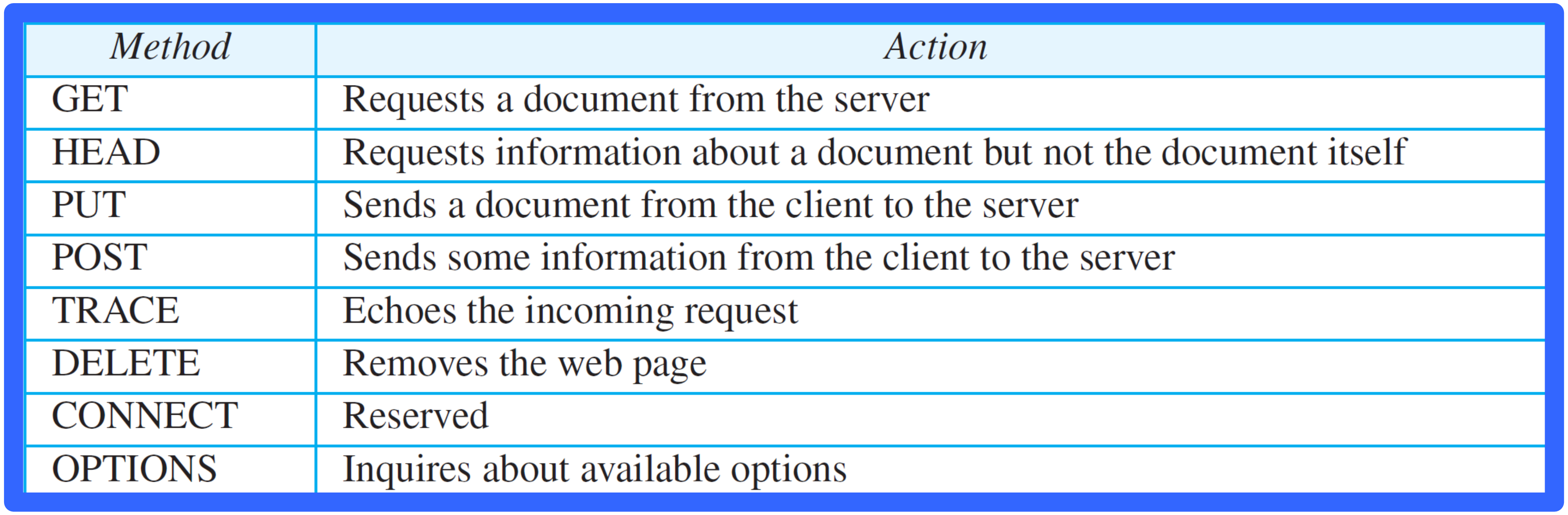
Response message



```
PUT   /cgi-bin/doc.pl  HTTP/1.1
Accept: */*
Accept: image/gif
Accept: image/jpeg
Content-length: 50

(Input information)
```

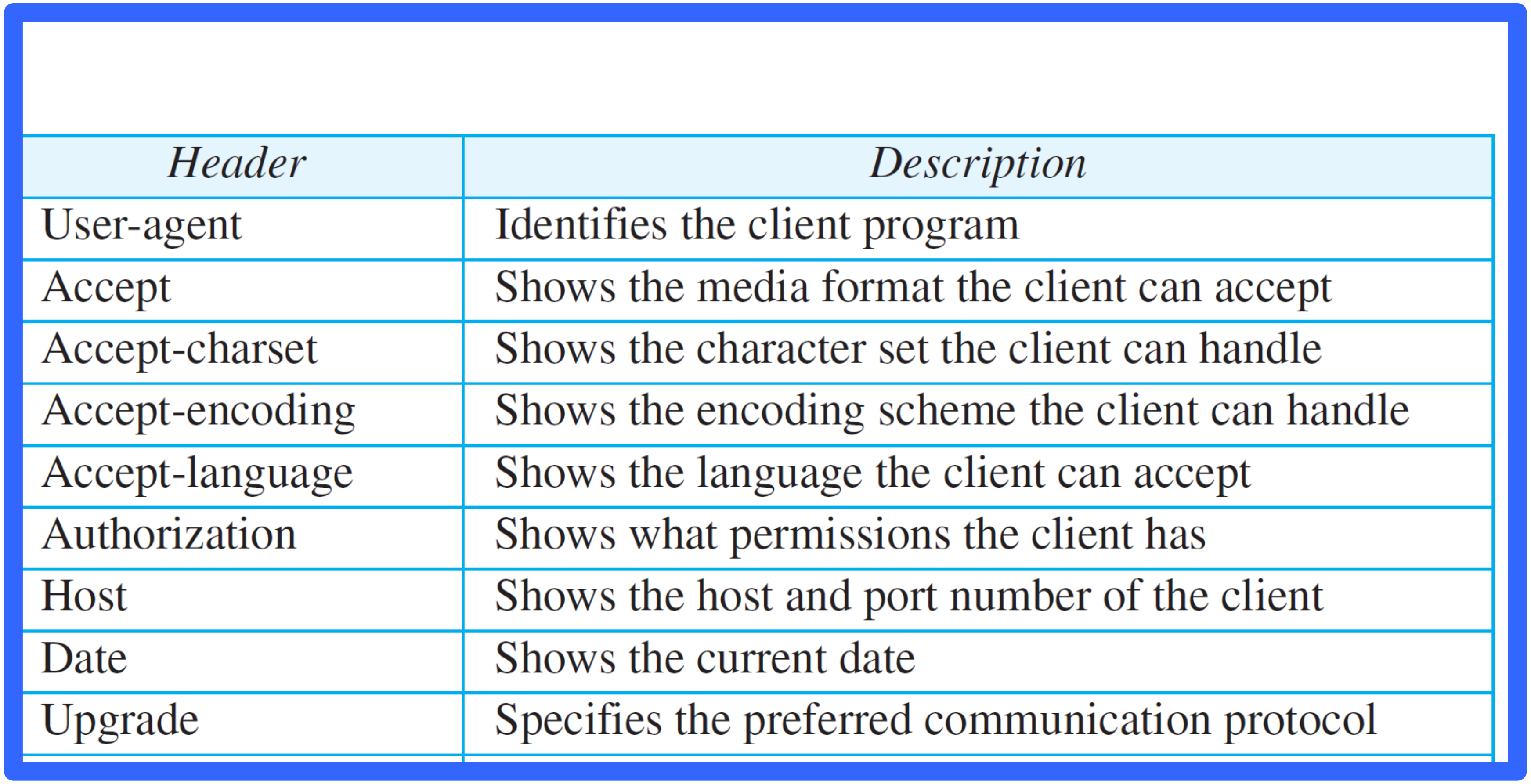
```
HTTP/1.1   200  OK
Date: Mon, 10-Jan-2011 13:15:14 GMT
Server: Challenger
Content-encoding: MIME-version 1.0
Content-length: 2000

(Body of the document)
```

Methods



Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
PUT	Sends a document from the client to the server
POST	Sends some information from the client to the server
TRACE	Echoes the incoming request
DELETE	Removes the web page
CONNECT	Reserved
OPTIONS	Inquires about available options

Request Header Names

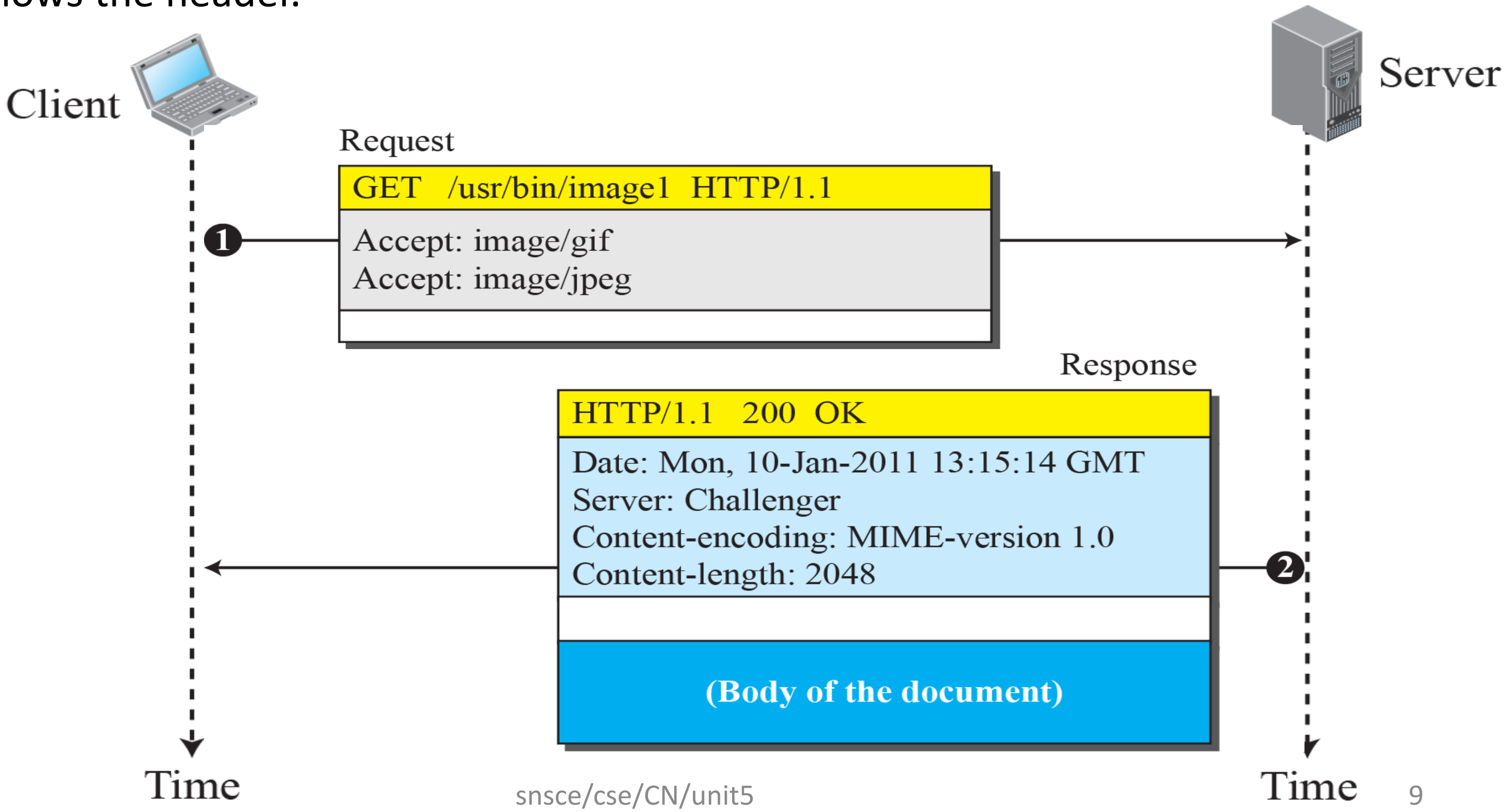


Header	*Description*
User-agent	Identifies the client program
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
Host	Shows the host and port number of the client
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol

Response Header Names

Header	*Description*
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Server	Gives information about the server
Set-Cookie	The server asks the client to save a cookie
Content-Encoding	Specifies the encoding scheme
Content-Language	Specifies the language
Content-Length	Shows the length of the document
Content-Type	Specifies the media type
Location	To ask the client to send the request to another site
Accept-Ranges	The server will accept the requested byte-ranges
Last-modified	Gives the date and time of the last change

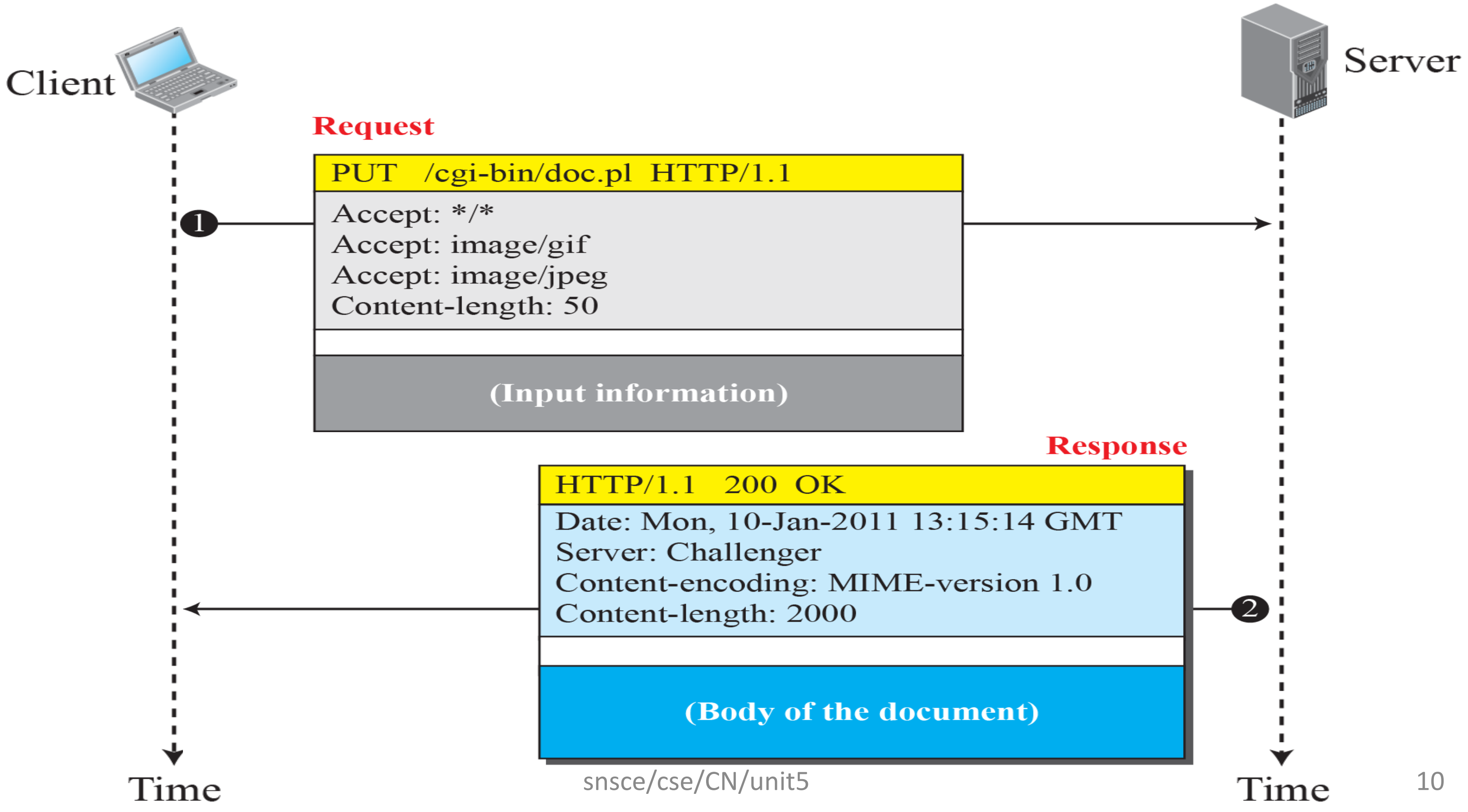
This example retrieves a document . We use the GET method to retrieve an image with the path /usr/bin/image1. The request line shows the method (GET), the URL, and the HTTP version (1.1). The header has two lines that show that the client can accept images in the GIF or JPEG format. The request does not have a body.
The response message contains the status line and four lines of header. The header lines define the date, server, content encoding and length of the document. The body of the document follows the header.



```
GET   /usr/bin/image1  HTTP/1.1
Accept: image/gif
Accept: image/jpeg
```

```
HTTP/1.1   200  OK
Date: Mon, 10-Jan-2011 13:15:14 GMT
Server: Challenger
Content-encoding: MIME-version 1.0
Content-length: 2048

(Body of the document)
```

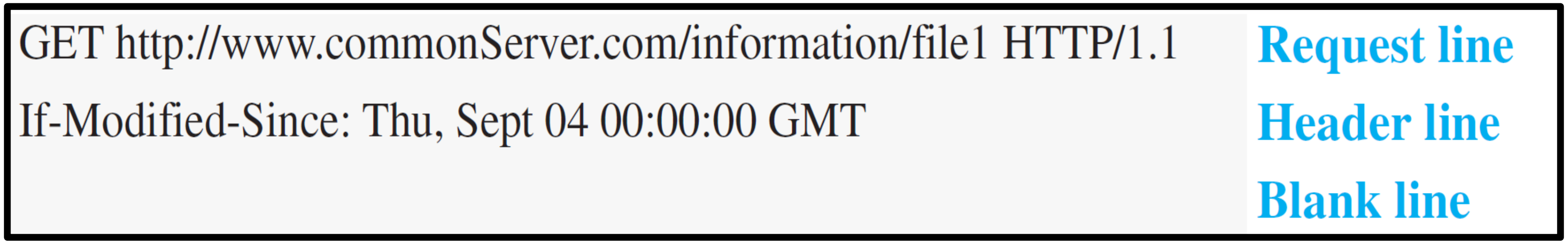
In this example, the client wants to send a web page to be posted on the server. We use the PUT method. The request line shows the method (PUT), URL, and HTTP version (1.1). There are four lines of headers. The request body contains the web page to be posted. The response message contains the status line and four lines of headers. The created document, which is a CGI document, is included as the body (see Figure 26.7).



```
PUT   /cgi-bin/doc.pl  HTTP/1.1
Accept: */*
Accept: image/gif
Accept: image/jpeg
Content-length: 50

(Input information)
```

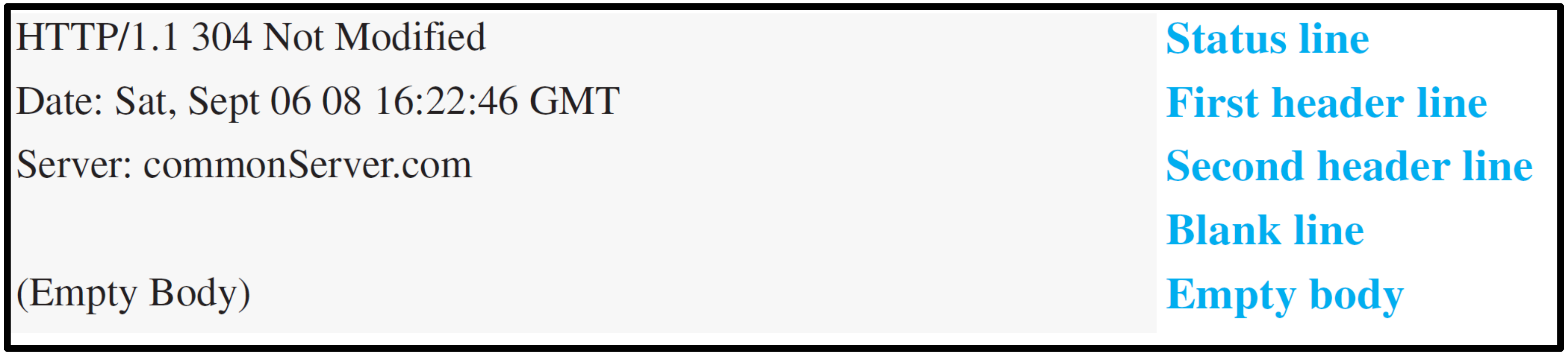
```
HTTP/1.1   200  OK
Date: Mon, 10-Jan-2011 13:15:14 GMT
Server: Challenger
Content-encoding: MIME-version 1.0
Content-length: 2000

(Body of the document)
```

The following shows how a client imposes the modification data and time condition on a request.



GET http://www.commonServer.com/information/file1 HTTP/1.1	Request line
If-Modified-Since: Thu, Sept 04 00:00:00 GMT	Header line
	Blank line

The status line in the response shows the file was not modified after the defined point in time. The body of the response message is also empty.



HTTP/1.1 304 Not Modified	Status line
Date: Sat, Sept 06 08 16:22:46 GMT	First header line
Server: commonServer.com	Second header line
	Blank line
(Empty Body)	Empty body

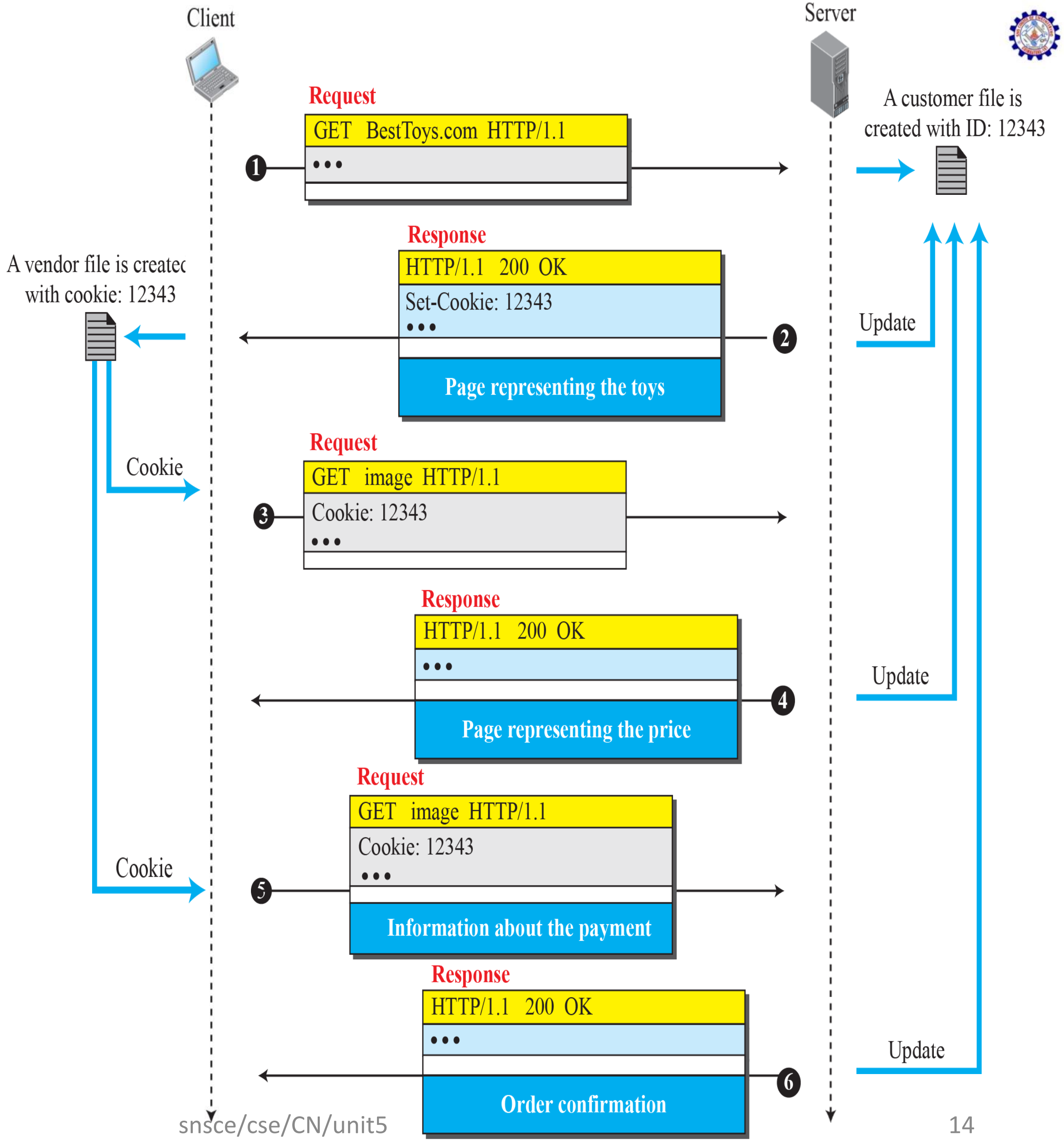
- **Cookies** are most commonly used to track website activity. When you visit some sites, the server gives you a **cookie** that acts as your identification card.
- Upon each return visit to that site, your browser passes that **cookie** back to the server. ... Servers can use **cookies** to provide personalized web pages

How to delete cookies

In Chrome

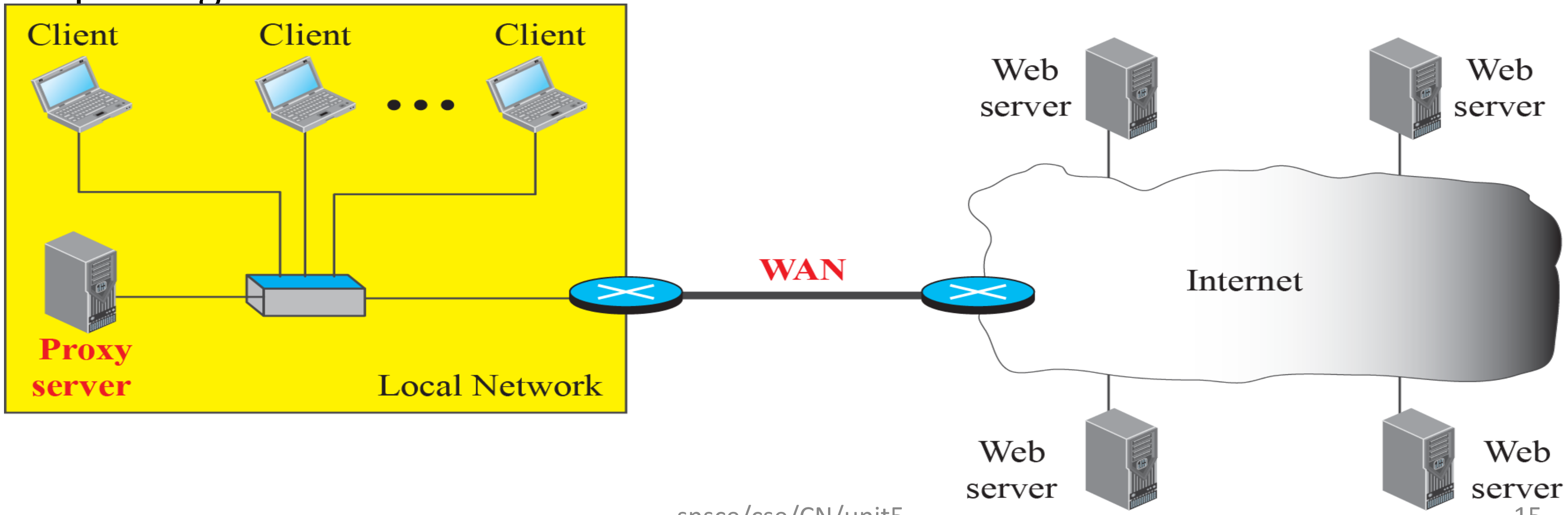
- On your computer, open Chrome.
- At the top right, click More .
- Click More tools
- Click Clear browsing data.
- At the top, choose a time range. To delete everything, select All time.
- Next to "Cookies and other site data" and "Cached images and files," check the boxes.
- Click Clear data.

Figure shows a scenario in which an electronic store can benefit from the use of cookies. Assume a shopper wants to buy a toy from an electronic store named BestToys.
The shopper browser (client) sends a request to the BestToys server.
 The server creates an empty shopping cart (a list) for the client and assigns an ID to the cart (for example, 12343).
The server then sends a response message, which contains the images of all toys available, with a link under each toy that selects the toy if it is being clicked.
 This response message also includes the Set-Cookie header line whose value is 12343.
 The client displays the images and stores the cookie value in a file named BestToys.



Web Caching: Proxy Servers: A proxy server is a computer that keeps copies of responses to recent requests.

Figure shows an example of a use of a proxy server in a local network, such as the network on a campus or in a company. The proxy server is installed in the local network. When an HTTP request is created by any of the clients (browsers), the request is first directed to the proxy server If the proxy server already has the corresponding web page, it sends the response to the client. Otherwise, the proxy server acts as a client and sends the request to the web server in the Internet. When the response is returned, the proxy server makes a copy and stores it in its cache before sending it to the requesting client.


Client
Client
Client
Proxy server
Local Network
WAN
Internet
Web server
Web server
Web server
Web server