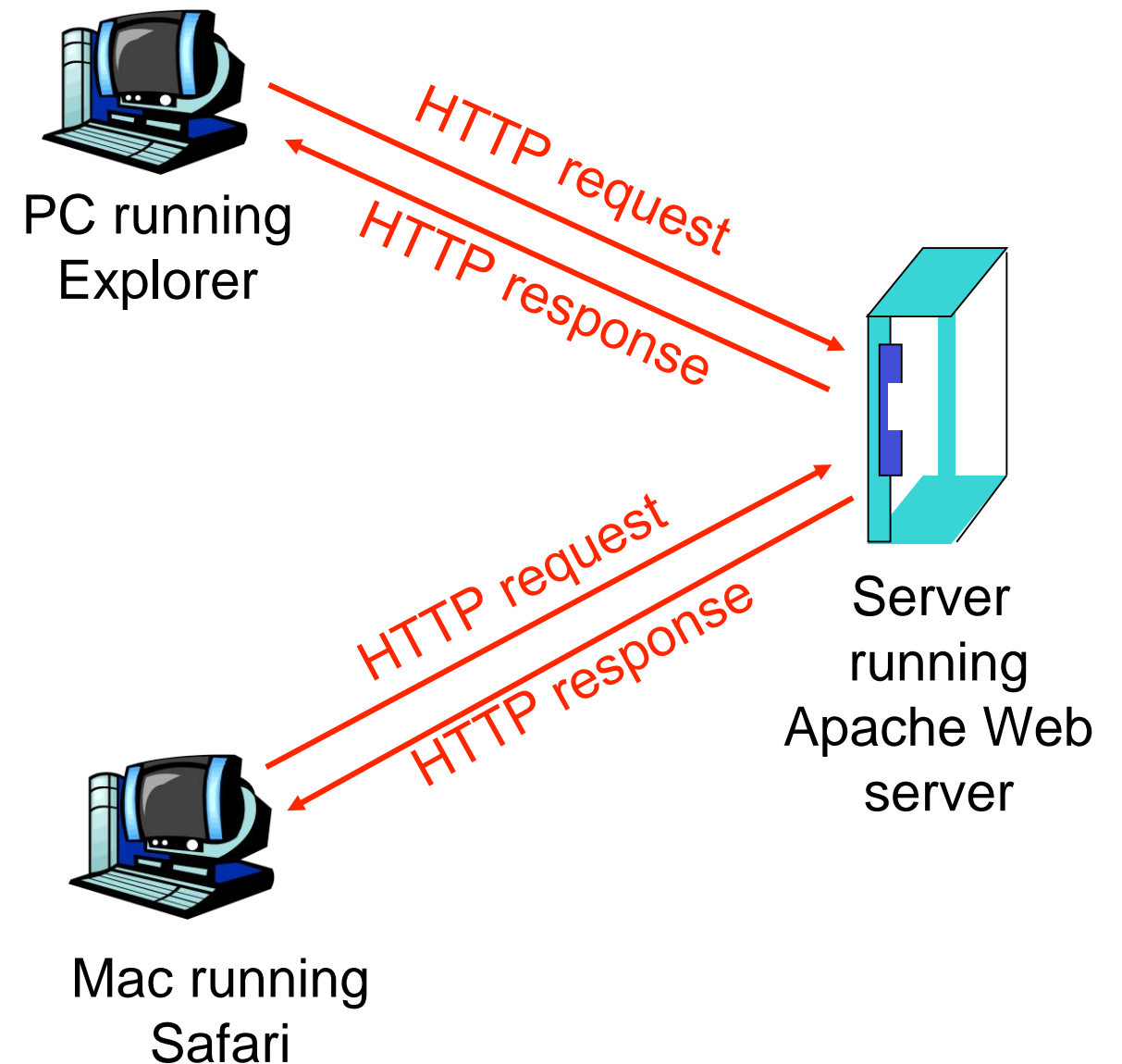


- HTTP: Hypertext Transfer Protocol
 - Anwendungsschicht-Protokoll des Webs
- Client/Server-Modell
 - Client
 - Browser fragt an
 - erhält und zeigt Web-Objekte an
 - Server
 - Web-Server sendet Objekte als Antwort der Anfrage

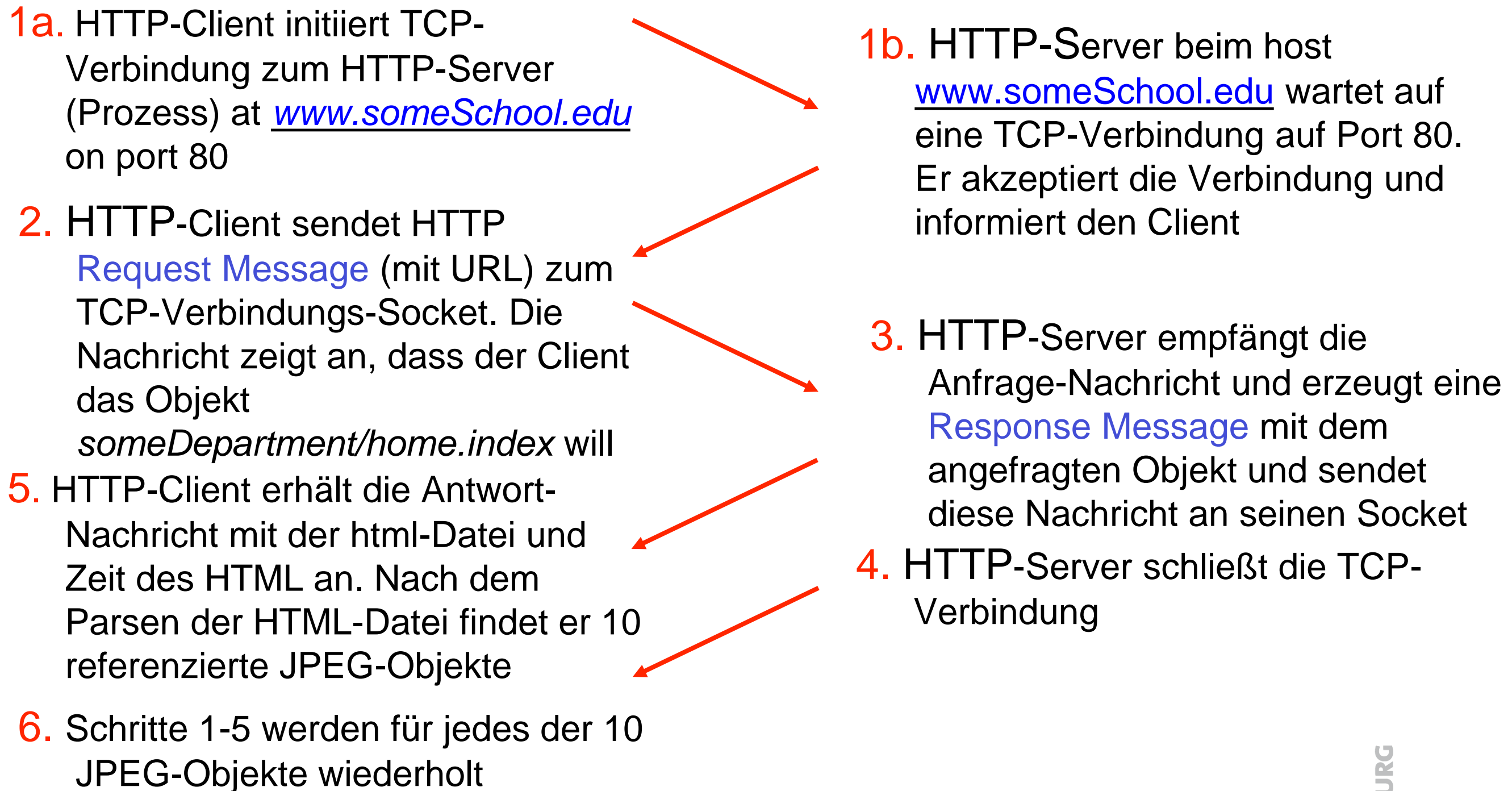


- Verwendet TCP
- Client initiiert TCP-Verbindung
 - erzeugt Socket zum Server auf Port 80
- Server akzeptiert TCP-Verbindung vom Client
- HTTP-Nachrichten
 - zwischen HTTP-Client und HTTP-Server
 - Anwendungsschicht-Protokoll-Nachrichten
- TCP-Verbindung wird geschlossen

- HTTP ist zustandslos (stateless)
 - Server merkt sich nichts über vorige Anfragen
- Warum?
 - Protokolle mit Zuständen sind komplex
 - Zustände müssen gemerkt und zugeordnet werden
 - falls Server oder Client abstürzen, müssen die möglicherweise inkonsistenten Zustände wieder angepasst werden

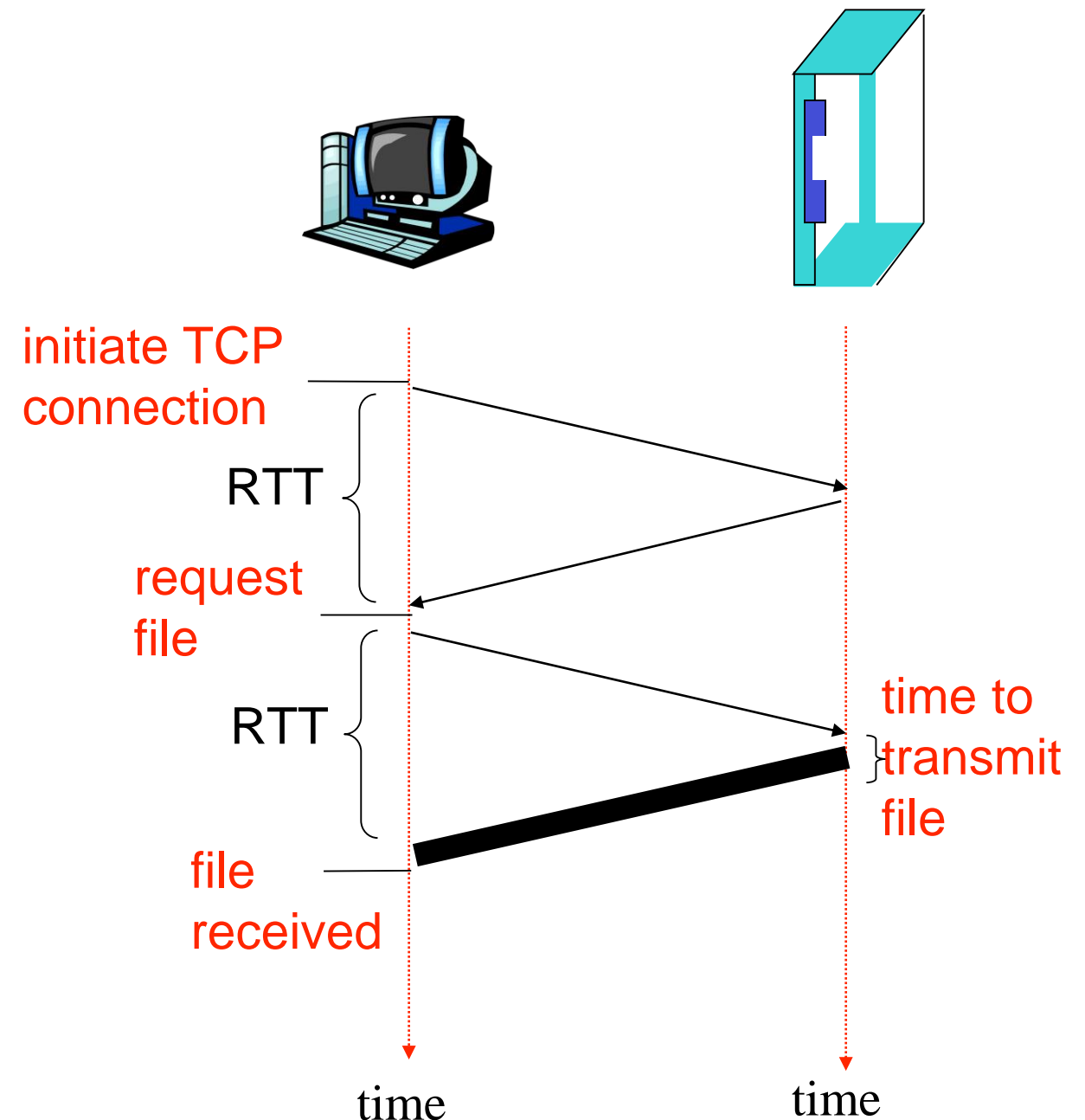
- Abbrechende (nicht persistente) HTTP-Verbindung
 - Höchstens ein Objekt wird über eine TCP-Verbindung gesendet
- Weiter bestehende (persistente) HTTP
 - Verschiedene Objekte können über eine bestehende TCP-Verbindung zwischen Client und Server gesendet werden

Nicht-Persistente HTTP-Verbindung

- 
- The diagram illustrates the sequence of steps for a non-persistent HTTP connection. It consists of two columns of text. The left column contains steps 1a, 2, 5, and 6. The right column contains steps 1b, 3, 4, and 5. Red arrows indicate the flow of the process: from 1a to 1b, from 2 to 3, from 5 to 4, and from 6 to 5. Step 5 in the left column is a summary of the repetition of steps 1-5 for multiple objects.
- 1a. HTTP-Client initiiert TCP-Verbindung zum HTTP-Server (Prozess) at www.someSchool.edu on port 80
 - 1b. HTTP-Server beim host www.someSchool.edu wartet auf eine TCP-Verbindung auf Port 80. Er akzeptiert die Verbindung und informiert den Client
 2. HTTP-Client sendet HTTP **Request Message** (mit URL) zum TCP-Verbindungs-Socket. Die Nachricht zeigt an, dass der Client das Objekt *someDepartment/home.index* will
 3. HTTP-Server empfängt die Anfrage-Nachricht und erzeugt eine **Response Message** mit dem angefragten Objekt und sendet diese Nachricht an seinen Socket
 4. HTTP-Server schließt die TCP-Verbindung
 5. HTTP-Client erhält die Antwort-Nachricht mit der html-Datei und Zeit des HTML an. Nach dem Parsen der HTML-Datei findet er 10 referenzierte JPEG-Objekte
 6. Schritte 1-5 werden für jedes der 10 JPEG-Objekte wiederholt

Nicht-persistentes HTTP: Antwortzeit

- Umlaufzeit (RTT – Round Trip Time)
 - Zeit für ein Packet von Client zum Server und wieder zurück
- Antwortzeit (Response Time)
 - eine RTT um TCP-Verbindung zu initiieren
 - eine RTT für HTTP Anfrage und die ersten Bytes des HTTP-Pakets
 - Transmit Time: Zeit für Dateiübertragung
- $\text{Zeit} = 2 \text{ RTT} + \text{transmit time}$



■ Nicht-persistentes HTTP

- benötigt 2 RTTs pro Objekt
- Betriebssystem-Overhead für jede TCP-Verbindung
- Browser öffnet oft TCP-Verbindungen parallel um referenzierte Objekte zu laden

■ Persistentes HTTP

- Server lässt die Verbindung nach der Antwortnachricht offen
- Folgende HTTP-Nachrichten zwischen den gleichen Client/Server werden über die geöffnete Verbindung versandt
- Client sendet Anfragen, sobald es ein referiertes Objekt findet
- höchstens eine Umlaufzeit (RTT) für alle referenzierten Objekte

HTTP-Request Nachricht

- Zwei Typen der HTTP-Nachricht: request, response
- HTTP-Request Nachricht:
 - ASCII (human-readable format)

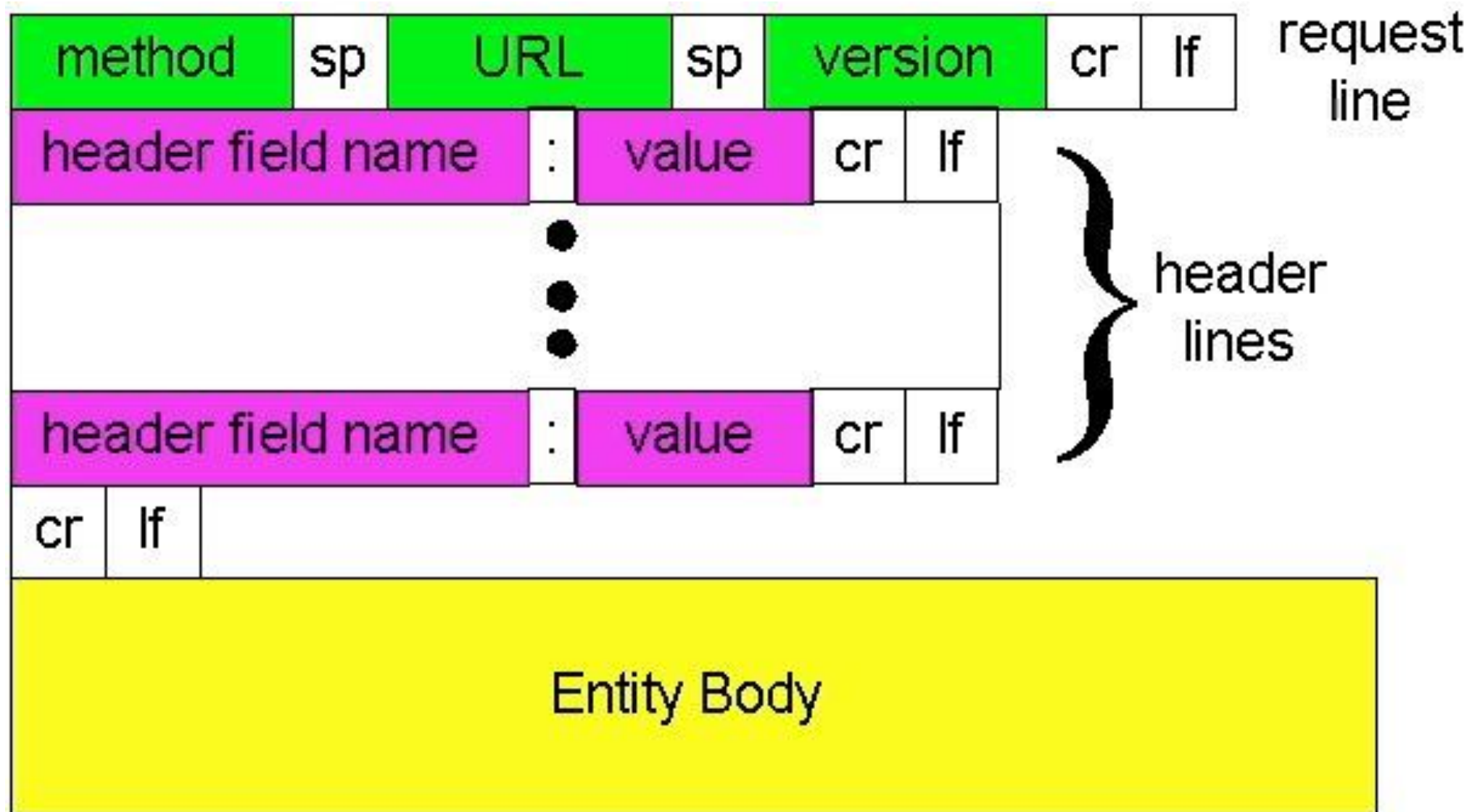
Request Zeile
(GET, POST,
HEAD Befehle)

```
GET /somedir/page.html HTTP/1.1  
Host: www.someschool.edu  
User-agent: Mozilla/4.0  
Connection: close  
Accept-language: fr
```

Extra Zeilenschaltung
zeigt das Ende der
Nachricht an

(extra carriage return, line feed)

HTTP-Request Nachricht: Allgemeines Format



■ Post

- Web-Seiten haben öfters Leerfelder für Eingaben
- Eingabe wird im *Body* zum Server hochgeladen

■ URL-Methode

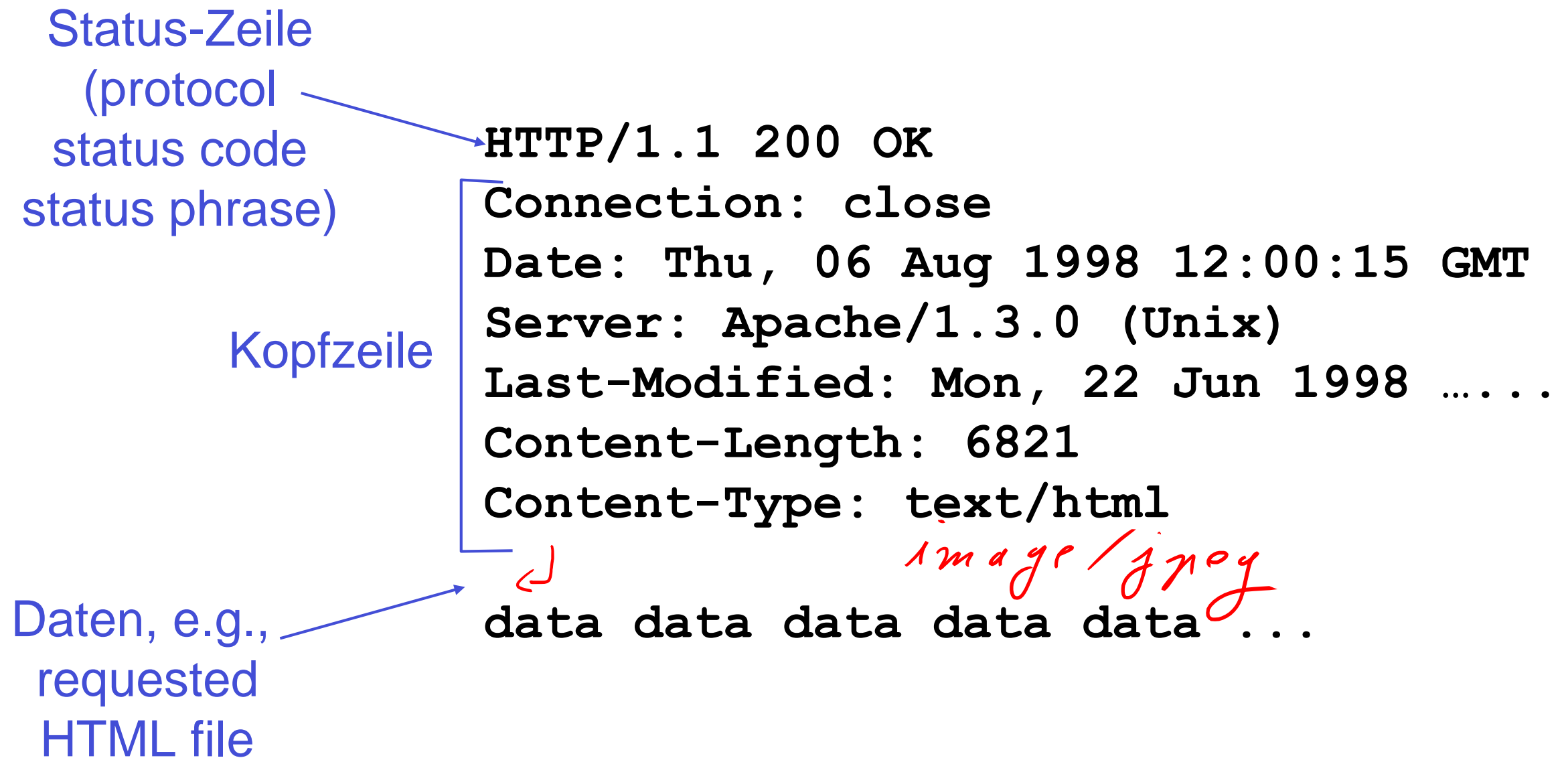
- Verwendet GET-Methode
- Input wird im URL-Feld der Anfrage-Nachricht gesendet:

`www.somesite.com/animalsearch?monkeys&banana`

... ? monkey = 15 & banana = 10

- HTTP/1.0
 - GET
 - POST
 - HEAD
 - fragt den Server nur nach dem Head, nicht nach dem Inhalt (*body*)
- HTTP/1.1
 - GET, POST, HEAD
 - PUT
 - lädt eine Datei im *body*-Feld zum Pfad hoch, der im URL-Feld spezifiziert wurde
 - DELETE
 - löscht Datei, die im URL-Feld angegeben wurde

HTTP-Antwort Nachricht



1. Telnet zum Web-Server

```
telnet cis.poly.edu 80
```

Öffnet TCP Verbindung auf Port 80
(default HTTP Server-Port) von cis.poly.edu.

2. Eingabe einer GET HTTP Anfrage:

```
GET /~ross/ HTTP/1.1  
Host: cis.poly.edu
```

Erzeugt einen minimalen
und vollständigen GET-Request
zu einem HTTP-Server

3. Was kommt als Antwort vom HTTP server?

HTTP Antwort-Status

- In der ersten Zeile der Client-Antwort-Nachricht (client response)
- Beispiele:
 - 200 OK
 - Anfrage wird beantwortet in dieser Nachricht
 - 301 Moved Permanently
 - neue Adresse für Objekt
 - Adresse folgt in der Nachricht
 - 400 Bad Request
 - Anfrage wird nicht verstanden
 - 404 Not Found
 - Angefragtes Dokument nicht vorhanden
 - 505 HTTP Version Not Supported

- Viele Web-Sites verwenden Cookies
- Vier Komponenten
 - 1) Cookie Kopf-Zeile der HTTP-Antwort-Nachricht (Response Message)
 - 2) Cookie-Kopf-Zeile in HTTP-Anfrage-Nachricht (Request Message)
 - 3) Cookie-Datei auf dem Benutzer-Rechner
 - wird vom Web-Browser des Benutzers unterhalten
 - 4) Datenbank auf der Web-Site (des Servers)

- Beispiel:
- Susan
 - surft das Web vom PC
 - besucht E-Commerce-Site *Amazon* zum ersten Mal
 - wenn die HTTP-Anfrage die Site erreicht, erzeugt die Web-Site
 - eindeutige ID
 - Eintrag in der Datenbank des Web-Servers

Cookies: Erzeugen einer Status-Information

*Set-cookie: sessionid=12345; expires = <datum> +zeit;
path=/forum; secure; HttpOnly*

Client

Server

