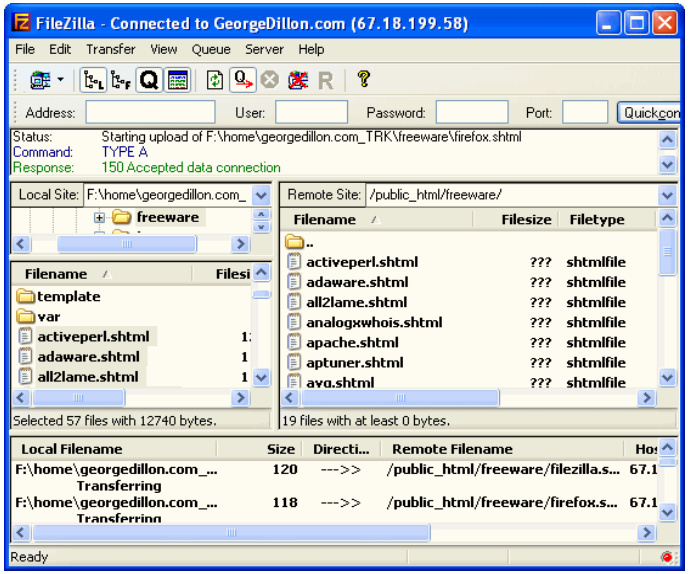
2. prednáška

Aplikačná vrstva

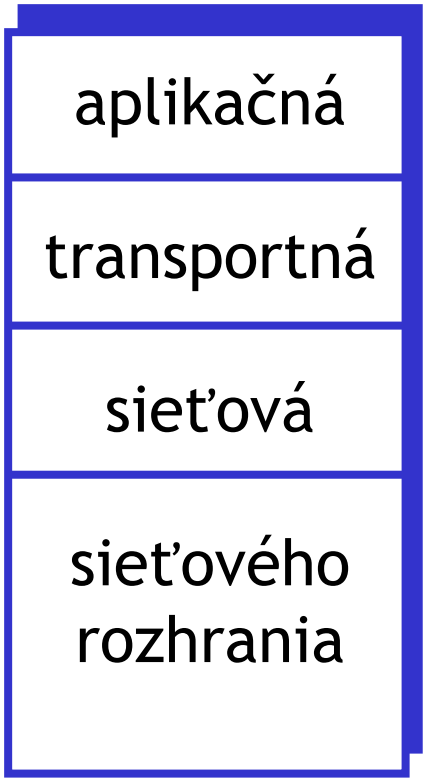


Prehľad prednášky

- úloha aplikačnej vrstvy
- využitie nižších vrstiev
- architektúry sieťových aplikácií
- aplikačné protokoly
  - HTTP
    - web
  - FTP
    - prenos súborov
  - SMTP, POP, IMAP
    - E-mail

Referenčný model TCP/IP

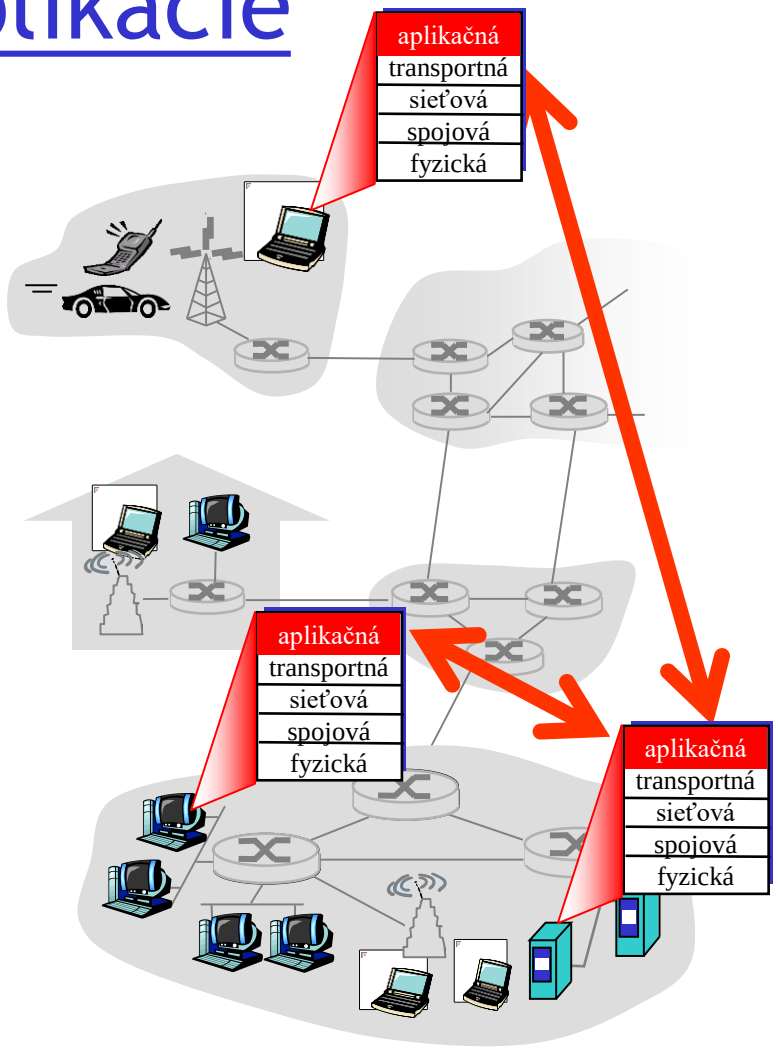
- aplikačná (application): umožňuje fungovanie sieťových aplikácií – definuje tvar a poradie správ
  - prezentačná a relačná splynuli s aplikačnou
  - tieto služby musí aj tak mať implementované aplikácia, ak to potrebuje
  - a čo ak nepotrebuje?
  - HTTP, FTP, SMTP, POP, IMAP, XMPP, SSH, ...
- transportná (transport): prenáša dáta medzi dvoma procesmi na rôznych koncových zariadeniach
  - TCP, UDP
- sieťová (network): smeruje datagramy od odosielateľa k príjemcovi
  - IP, smerovacie protokoly
- sieťového rozhrania (network interface): splynutie funkcionality do technológií na prenos dát medzi susednými sieťovými prvkami a spôsobu prenášania binárnych dát
  - PPP, Ethernet



Vytváranie sieťovej aplikácie

programy

- bežia na (rôznych) *koncových systémoch*
- komunikujú prostredníctvom siete
- napr. web server komunikuje s browserom
- zariadenia jadra siete nespúšťajú používateľské aplikácie



Komunikácia procesov

Proces: program bežiaci na počítači

- V rámci jedného počítača procesy obvykle komunikujú medziprocesovou komunikáciou (definovanou v OS).
- procesy na rôznych počítačoch komunikujú prostredníctvom správ

Klientský proces: proces ktorý inicializuje komunikáciu

Serverový proces: proces, ktorý čaká na to, že sa naňho niekto napojí

- Poznámka: aj aplikácie vo všetkých P2P architektúrach majú klientské a serverovské procesy

Protokol aplikačnej vrstvy definuje:

- Typy správ, ktoré sa vymieňajú,
  - Napr. požiadavky, odpovede
- Syntax správ:
  - Z čoho sa skladajú jednotlivé správy, rozsahy hodnôt
- Sémantika správ
  - Význam správ a informácií v nich
- Pravidlá pre to, kedy, za akých okolností, a ako si budú procesy posielať správy

Verejné protokoly:

- Definované v RFC
- Umožňujú interoperabilitu
- napr. HTTP, SMTP

Proprietárne protokoly:

- napr. Skype

Prehľad prednášky

- úloha aplikačnej vrstvy
- využitie nižších vrstiev
- architektúry sieťových aplikácií

- aplikačné protokoly
  - HTTP
    - web
  - FTP
    - prenos súborov
  - SMTP, POP, IMAP
    - E-mail

Adresácia procesov (sieťová vrstva)

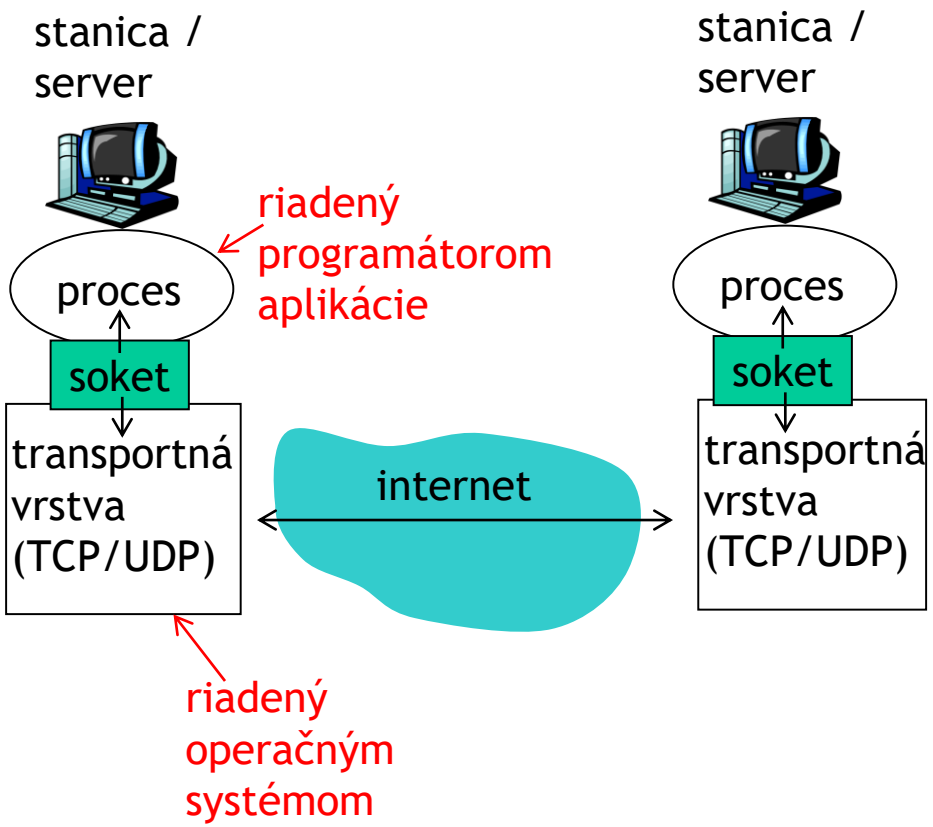
- aby sa procesu dali posielať správy, musíme ho na internete presne identifikovať
- počítač musí mať jedinečnú IP adresu
- stačí IP adresa počítača na identifikáciu procesu sieťovej aplikácie?

Adresácia procesov (sieťová a transportná vrstva)

- aby sa procesu dali posielať správy, musíme ho na internete presne identifikovať
- počítač musí mať jedinečnú IP adresu
- stačí IP adresa počítača na identifikáciu procesu sieťovej aplikácie?
  - Nie! Na jednom počítači môže byť viac sieťových procesov
- *identifikátor* zahŕňa IP adresu a číslo portu(portov) priradené k procesu na počítači.
- Príklady čísiel portov:
  - HTTP server: 80
  - Mailový server: 25
- Na poslanie HTTP správy na web server ics.upjs.sk potrebujeme:
  - IP adresu: 158.197.31.29
  - Číslo portu: 80

Komunikácia s transportnou vrstvou - Sokety

- procesy posielajú a prijímajú správy prostredníctvom svojich soketov
- Čo je soket?
  - rozhranie, cez ktoré komunikuje sieťová aplikácia s implementáciou protokolu transportnej vrstvy v operačnom systéme



API: (1) iné pre rôzne transportné protokoly

(2) umožňuje nastaviť niektoré parametre nižších vrstiev

Akú transportnú službu si vybrať pre aplikáciu?

Strata dát

- niektoré aplikácie (napr. vysielanie on-line kamery) sú schopné tolerovať nejakú stratu dát
- iné aplikácie (napr. prenos súborov, SSH) vyžadujú 100% spoľahlivé dátové prenosy

Časová tolerancia

- niektoré aplikácie (napr., telefonovanie, hry) vyžadujú malé zdržania prenosu

Šírka pásma

- niektoré aplikácie nevyužijú veľkú šírku pásma
- iné využijú všetku šírku pásma, ktorú môžu

Požiadavky aplikácií na transportnú vrstvu

Aplikácia	Strata dát	Šírka pásma	Citlivosť na čas
prenos súborov	žiadna	prispôsobiteľná	nie
e-mail	žiadna	prispôsobiteľná	nie
Web	žiadna	prispôsobiteľná	nie
real-time audio/video	možná	audio: 5kb/s-1Mb/s video:10kb/s-5Mb/s	áno
uložené audio/video	možná	to isté	áno
interaktívne hry	možná	zopár kb/s	áno
instant messaging	žiadna	prispôsobiteľná	áno aj nie

Služby transportných protokolov

TCP služby:

- *so spojením:* vyžaduje nastavenie spojenia s príjemcom pred odosielaním prvých dát
- *potvrdzovaný prenos dát* medzi odosielajúcim a prijímajúcim procesom
- *kontrola toku dát* - odosielateľ nezahltí príjemcu
- *kontrola zahltenia siete* - odosielateľ spomalí odosielanie pri zahltení siete
- *neposkytuje* garantovanie času doručenia a minimálnej šírky pásma

UDP služby:

- *nepotvrdzovaný prenos dát* s "najväčším úsilím" medzi odosielajúcim a prijímajúcim procesom
- *komunikácia one-to-many (broadcast, multicast)*
- *real-time prenos – žiadne čakanie na chýbajúce dáta*
- *neposkytuje* potvrdzovanie dát, kontrolu toku dát, kontrolu zahltenia siete, garantovanie času doručenia a minimálnej šírky pásma

Aplikačné a transportné protokoly využívané sieťovými aplikáciami

Aplikácia	Protokol aplikačnej vrstvy	Transportný protokol čo využíva
e-mail	SMTP [RFC 2821]	TCP
vzd. prístup na konzolu	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
prenos súborov	FTP [RFC 959]	TCP
streamovanie multimédií	vlastné (napr. RealNetworks)	TCP alebo UDP
telefonovanie cez net	VoIP, vlastné (napr., Vonage,Dialpad)	typicky UDP
preklad doménových mien a IP adries	DNS	UDP

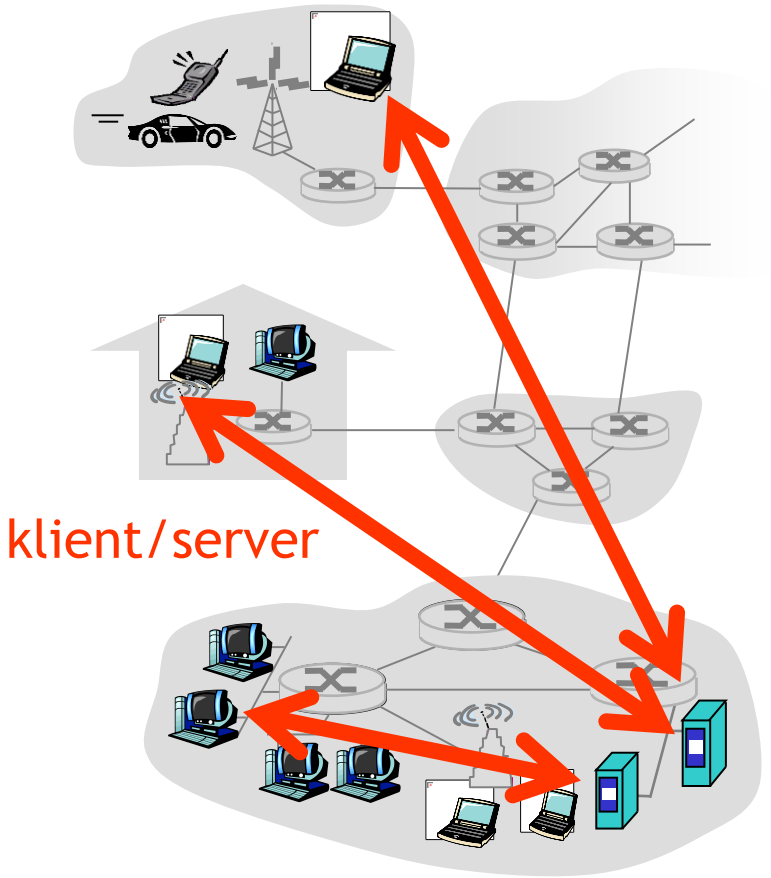
Prehľad prednášky

- úloha aplikačnej vrstvy
- využitie nižších vrstiev
- architektúry sieťových aplikácií
- aplikačné protokoly
  - HTTP
    - web
  - FTP
    - prenos súborov
  - SMTP, POP, IMAP
    - E-mail

Architektúry sieťových aplikácií

- Klient-server
- Peer-to-peer (P2P)
- Hybrid oboch

Architektúra klient-server



server:

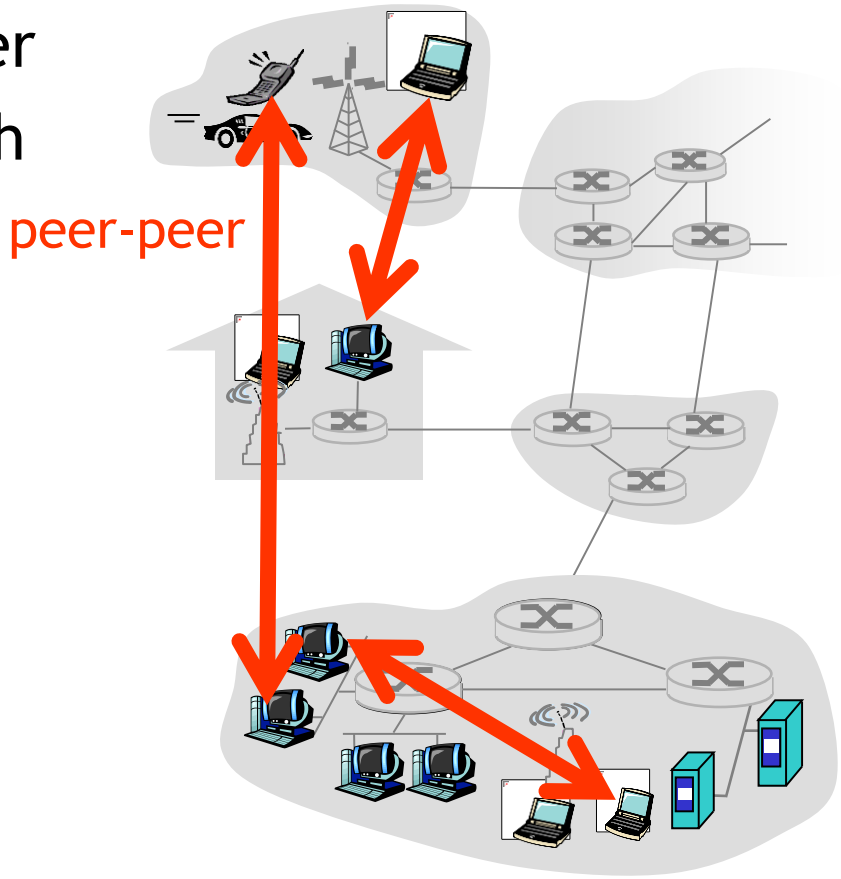
- stále zapnutý počítač
- pevná IP adresa
- serverové farmy pre lepšiu škálovateľnosť

klienti:

- komunikujú so serverom
- môžu sa priebežne odpájať
- môžu mať dynamickú IP adresu
- nekomunikujú medzi sebou

Čistá P2P architektúra

- *nemá* stále zapnutý server
- na ľubovoľných koncových systémoch
- komunikujú medzi sebou
- peerovia sú prerušovane napojení a môžu meniť IP adresy
- príklad: Gnutella



Vysoko škálovateľné, ale ťažko manažovateľné

Hybrid Klient-servera a P2P

Skype

- program na telefonovanie cez IP
- centrálny server: hľadá adresu cieľového používateľa cez klient-server
- komunikácia už typu peer-peer: priame spojenie (nie cez server)

Instant messaging

- chat medzi dvoma používateľmi ako P2P
- centralizovane sa zisťuje pripojenie klientov
  - klienti si v centrálnom serveri registrujú svoje IP adresy, keď sa prihlásia
  - klienti kontaktujú server, aby zistili IP adresy kontaktov, s ktorými chcú rozprávať

Prehľad prednášky

- úloha aplikačnej vrstvy
- využitie nižších vrstiev
- architektúry sieťových aplikácií

- aplikačné protokoly
  - HTTP
    - web
  - FTP
    - prenos súborov
  - SMTP, POP, IMAP
    - E-mail

Web a HTTP

označenia

- Webová stránka sa skladá z objektov
- Objekt môže byť HTML súbor, obrázok, JavaScriptový súbor, audio súbor, CSS súbor…
- Webovú stránku tvorí základný HTML súbor, ktorý obsahuje niekoľko odkazov na ďalšie objekty
- Adresa objektu je URL
- Príklad URL:

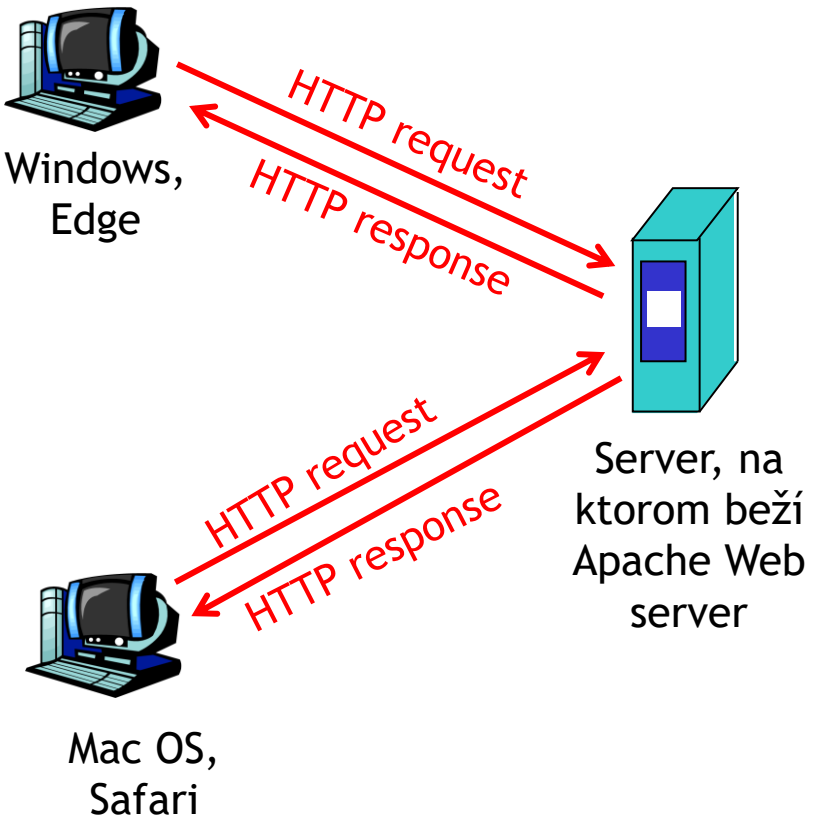
`www.inštitúcia.sk/oddelenie/obr.gif`

doménové meno — cesta

Náhľad na HTTP

HTTP: hypertext transfer protocol

- Protokol aplikačnej vrstvy pre Web
- klient/server model
  - *klient:* prehliadač, ktorý žiada, dostáva a zobrazuje webové objekty
  - *server:* webový server odosiela objekty v odpovediach na požiadavky
- HTTP 1.0: RFC 1945 (1996)
- HTTP 1.1: RFC 2068 (1997)
- HTTP 2.0: RFC 7540 (2015)



Náhľad na HTTP

Používa TCP:

- klient inicializuje TCP spojenie (vytvorí soket) so serverom na porte 80
- server akceptuje TCP spojenie od klienta
- browser (HTTP klient) a webový server (HTTP server) si posielajú HTTP správy (správy aplikačného protokolu)
- TCP spojenie sa zavrie

HTTP je "bezstavový"

- server si neuchováva históriu predchádzajúcich požiadaviek klienta

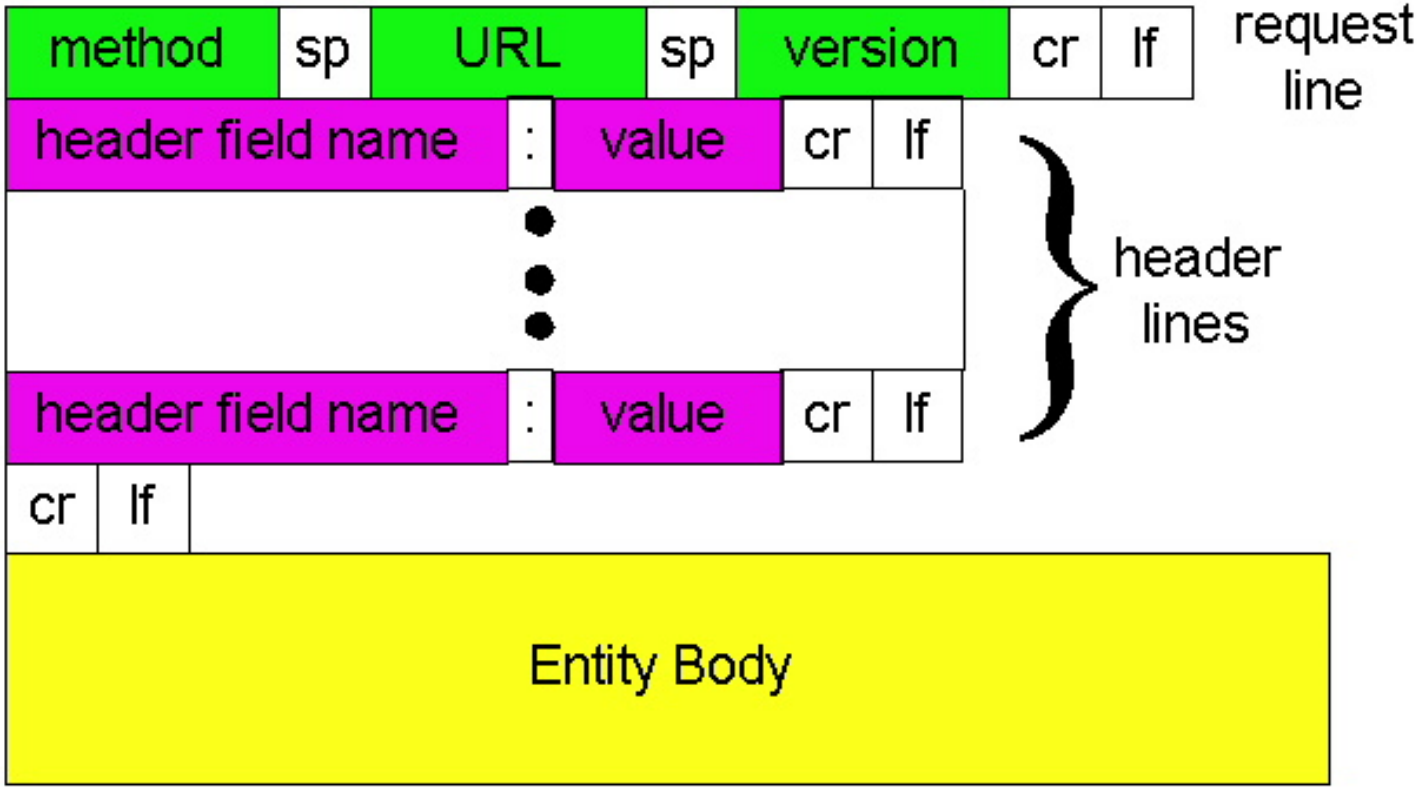
Stavové protokoly sú zložité!

- musí sa spravovať história
- po zlyhaní klienta alebo servera sa ich pohľad na to, čo je to posledný stav, môže líšiť a musia sa synchronizovať

HTTP/1.1 požiadavka

- Dva typy HTTP správ: *požiadavka(request)* a *odpoveď(response)*
- HTTP/1.1 požiadavka:
  - ASCII (ľudsky čitateľný tvar)

HTTP/1.1 požiadavka všeobecne



Odosielanie formulárových dát

POST:

- Pri odosielaní formulárov z webstránok lepšia alternatíva ku GET
  - URL má obmedzenú veľkosť
  - Dáta sú posielané na server v “entity body” - nevidieť dáta v adrese prehliadača

Spôsob cez URL:

- Používa príkaz GET
- Dáta sú súčasťou URL adresy:

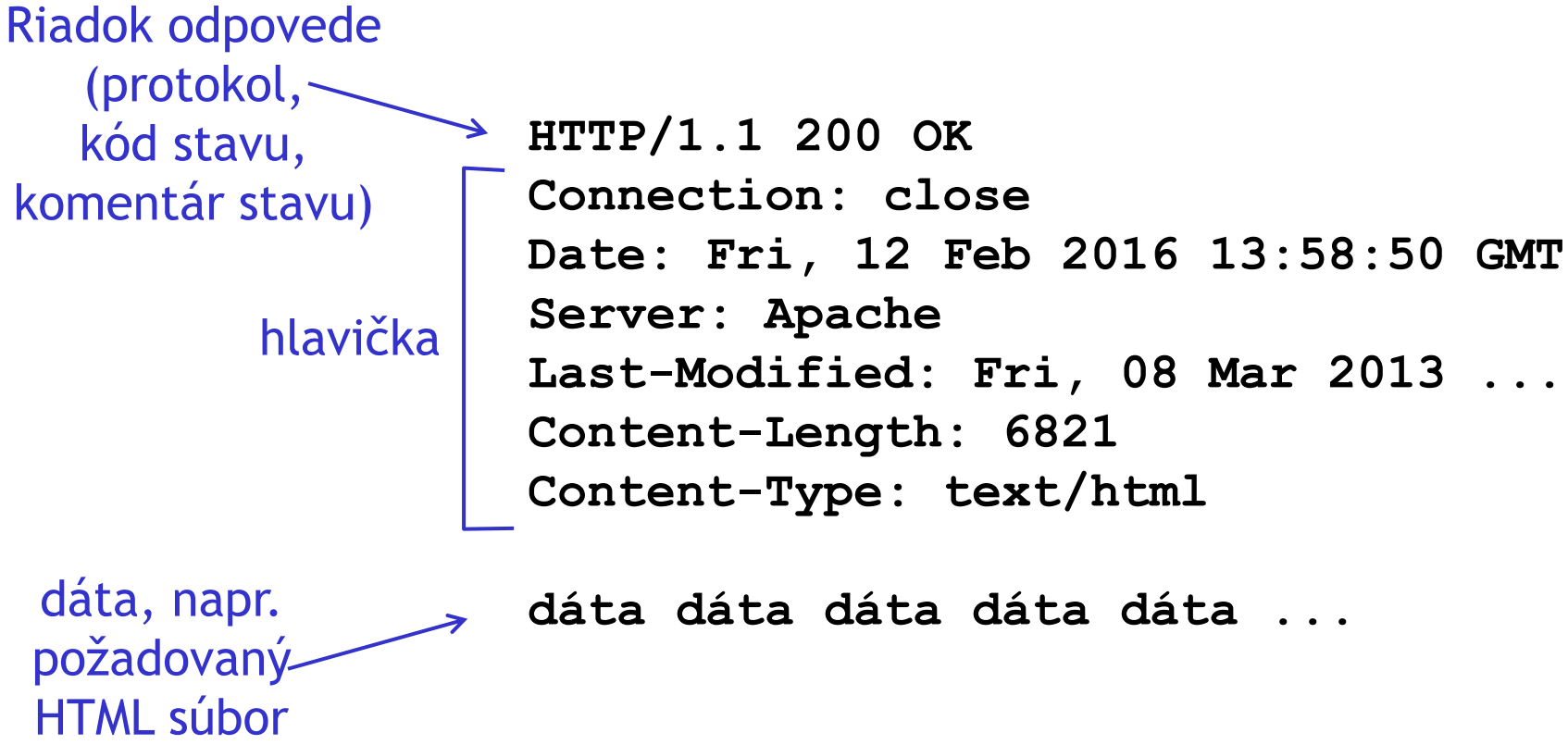
```
www.somesite.com/animalsearch?key1=val1&key2=val2
```

Príkazy požiadaviek

- GET, POST
- HEAD
  - "neposielaj žiadaný objekt, iba hlavičku"
- PUT
  - posiela súbor v "entity body" na objekt špecifikovaný v URL
- DELETE
  - vymaže objekt na adrese špecifikovanej v URL

Využitie v CRUD protokoloch napr. REST

HTTP/1.1 odpoveď



Stavy HTTP odpovede

Niektoré príklady prvého riadka odpovede servera:

200 OK

- požiadavka úspešná, požadovaný objekt nasleduje pod hlavičkou

301 Moved Permanently

- požadovaný objekt je presunutý, nová pozícia je špecifikovaná v hlavičke v časti “Location:”

400 Bad Request

- server nerozumie požiadavke

404 Not Found

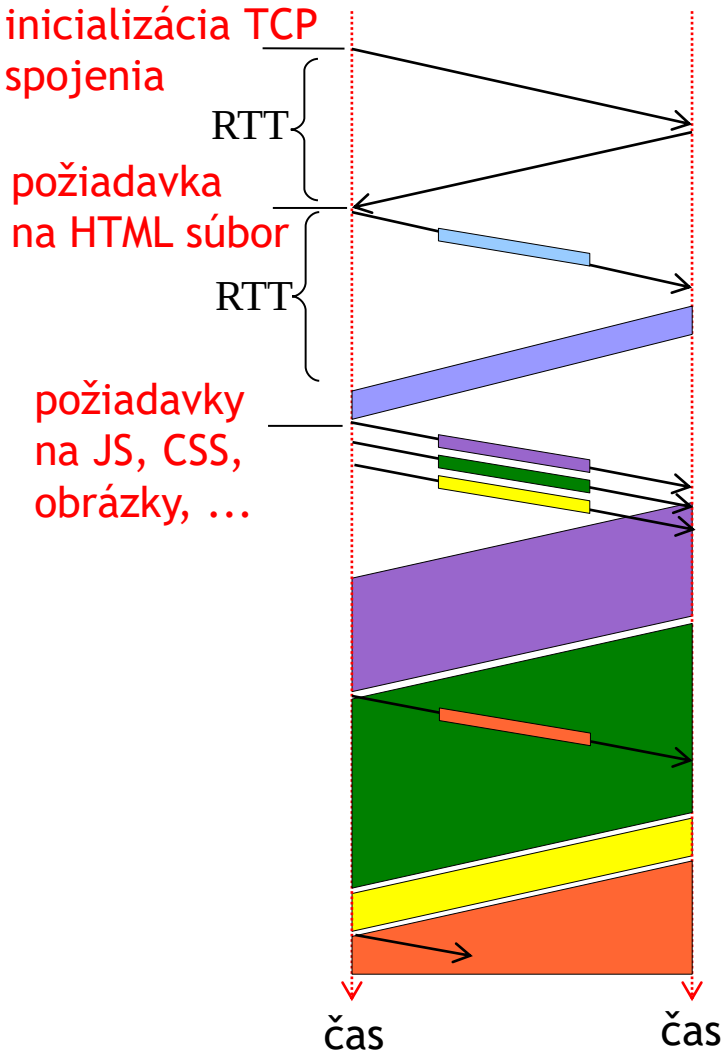
- požadovaný objekt sa nenašiel

505 HTTP Version Not Supported

- server nepodporuje požadovanú verziu HTTP protokolu

HTTP/1.1 spojenia

- Perzistentné spojenie
  - Nechám spojenie otvorené, pokiaľ nenatiahnem všetky objekty.
- Pipelining (reťazenie požiadaviek a odpovedí)
  - Po jednej požiadavke nečakám na odpoveď, ale zašlem všetky požiadavky za sebou.



Vyskúšajte si HTTP komunikáciu ako klient

1. Telnetujte svoj obľúbený web server:

```
telnet siete.gursky.sk 80
```

Otvorí TCP spojenie na porte 80 (defaultný port HTTP servera) na počítači siete.gursky.sk
Každý riadok odchádza ako správa na server siete.gursky.sk procesu s prideleným portom 80

2. Napíšte GET požiadavku:

```
GET / HTTP/1.1
Host: siete.gursky.sk
```

Po napísaní pridajte ešte prázdny riadok, čím ukončíte HTTP požiadavku minimálneho rozsahu

3. Môžete obdivovať odpoveď web servera

HTTP/2

- používanie binárnych rámcov namiesto pôvodných textových HTTP požiadaviek a odpovedí (všetky majú rovnaký tvar 9 bajtovej hlavičky)
- Proces požiadavky a zasielania jedného objektu, ktorý chceme stiahnuť zo servera, sa nazýva prúd a má pridelené jedinečné ID
- Objekty sa nemusia posielať celé naraz, ale môžu byť rozdelené do viacerých rámcov a zasielané na striedačku s rámcami iných prúdov. Všetky prúdy sú posielané cez to isté spojenie.
- 10 typov rámcov (SETTINGS, HEADERS, DATA, CONTINUATION,...)
- HEADERS rámec požiadavky, ako náprotivok hlavičky HTTP/1.x požiadavky, definuje nový prúd dát v rámci spojenia, ktorý má zabezpečiť stiahnutie jedného objektu.

HTTP/2

- každý prúd dát môže mať rôznu prioritu, ktorá umožní zasielanie rámcov jedného objektu častejšie ako iného
- nastavovanie vzájomných závislostí prúdov (ak mi nepošleš tento objekt, tak objekty na ňom závislé tiež neposielaj)
- kompresia hlavičiek, kde ďalšie hlavičky obsahujú iba rozdiely oproti predchádzajúcim hlavičkám
- možnosť pre server zasielať objekty bez vyžiadania, ak server predpokladá, že prehliadač bude daný objekt potrebovať
- možnosť použitia spojenia so serverom pre viaceré domény, ak sú hostované na serveri s rovnakou IP adresou

Niektoré HTTP/2 rámce

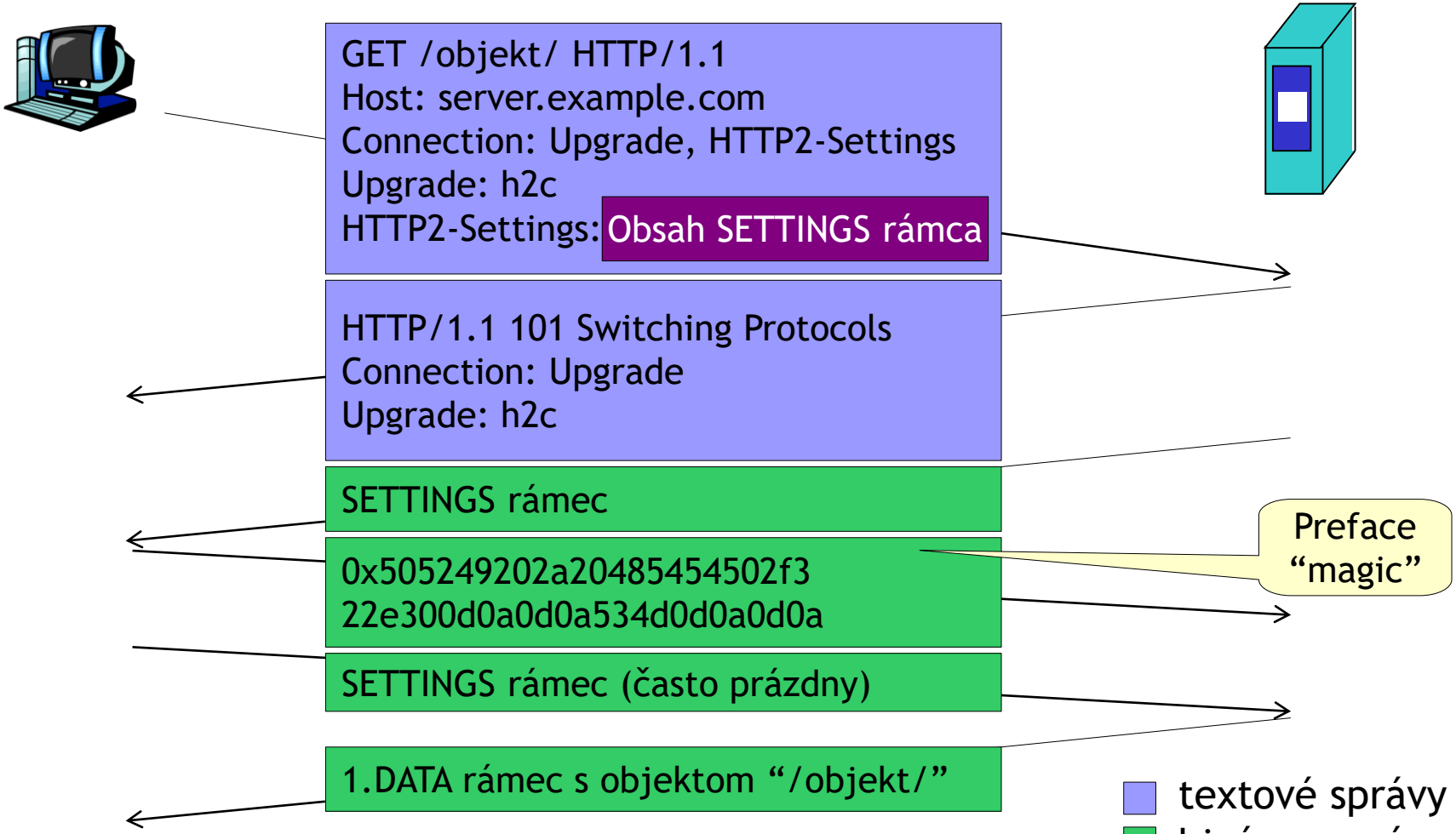
Spoločná hlavička:

Bit	+0..7	+8..15	+16..23	+24..31
0	Length			Type
32	Flags			
40	R	Stream Identifier		
...	*Frame Payload*			

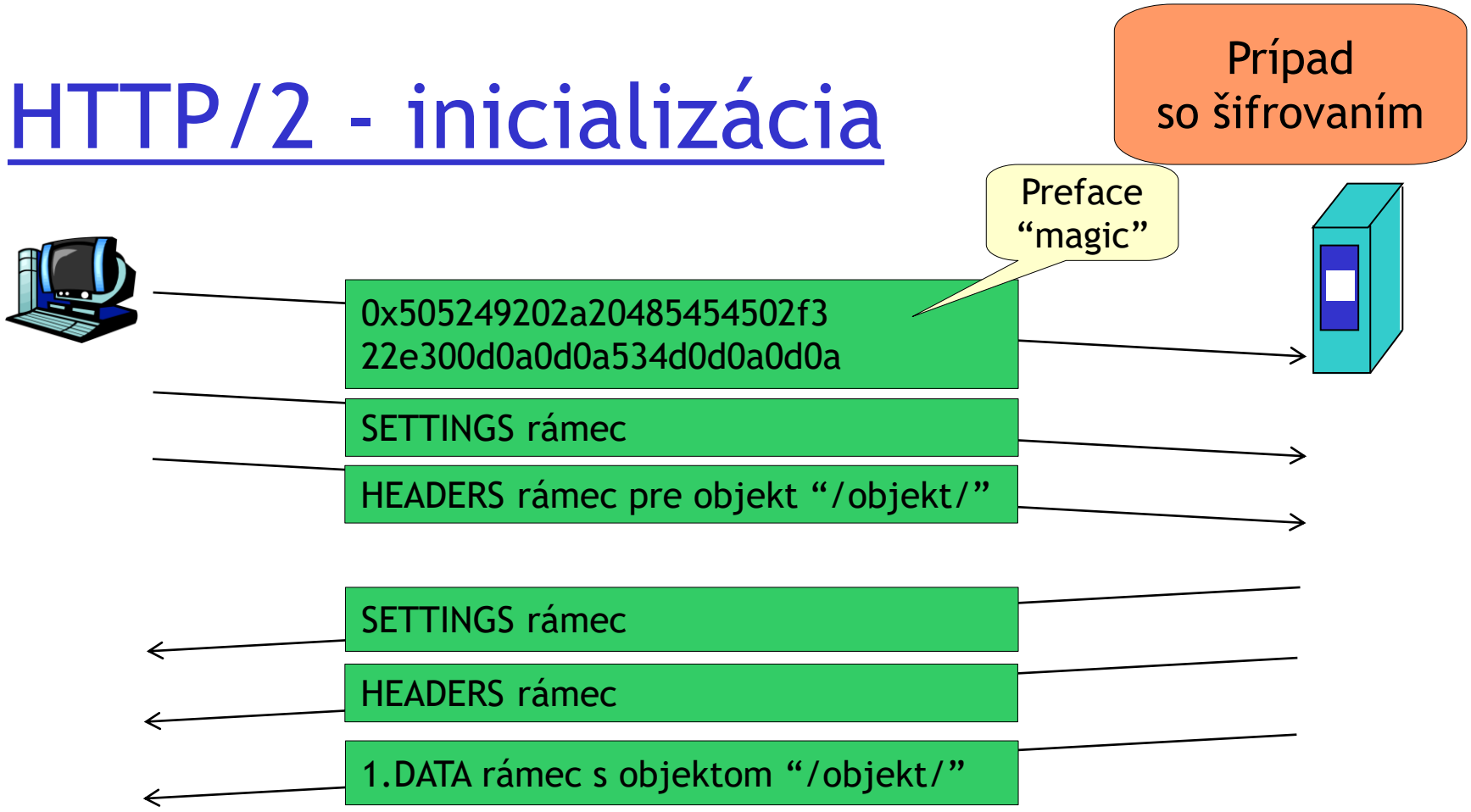
- SETTINGS – nastavuje parametre komunikácie - veľkosť rámcov, buffra (okna), max. počet prúdov,...
- HEADERS – prenáša hlavičku + prideľuje požadovanému objektu identifikátor prúdu
- DATA – prenáša telá HTTP správ
- CONTINUATION – pokračovanie dát daného prúdu
- PRIORITY – informuje o priorite prúdov
- PUSH_PROMISE – zahájenie nového prúdu zo strany servera (odporúčaný objekt)

HTTP/2 - inicializácia

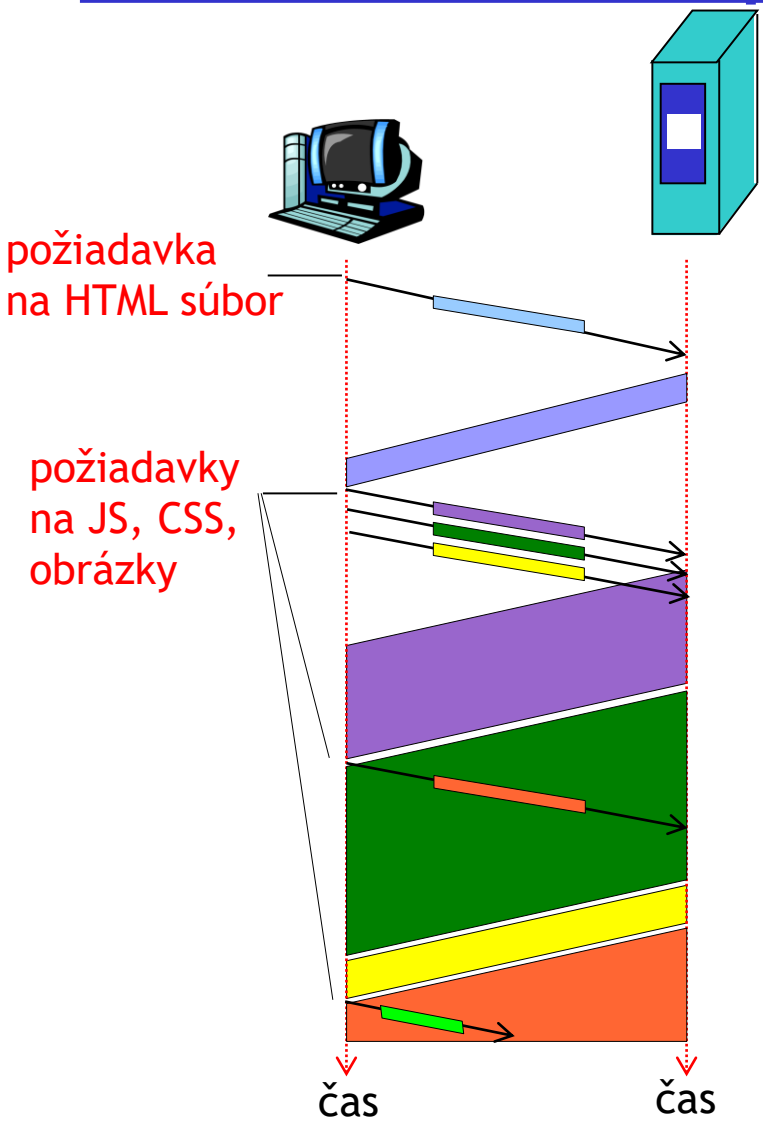
Prípad bez šifrovania

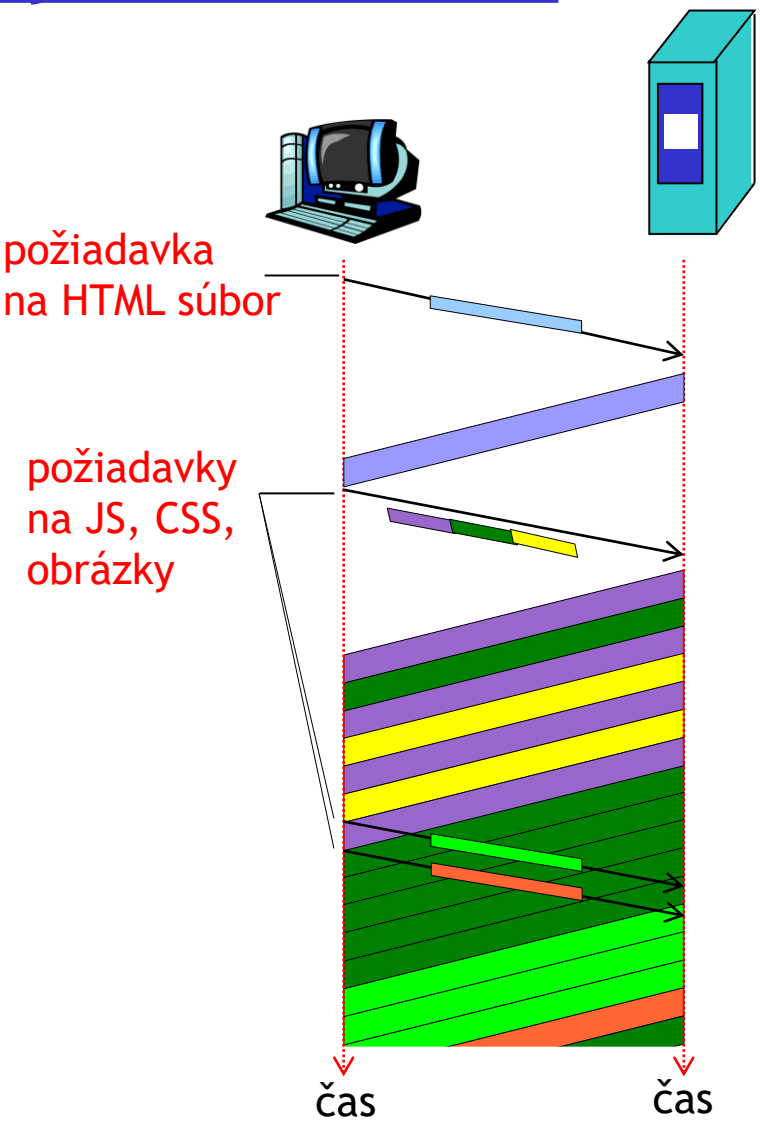


HTTP/2 - inicializácia



HTTP/1.1 vs. prúdy v HTTP/2





HTTP/2: Kompresia hlavičiek

Chcem poslať:

:method	GET
:scheme	https
:host	siete.gursky.sk
:path	/obr.jpg
user-agent	Mozilla/5.0...
content-type	image/jpg

Predchádzajúce:

:method	GET
:scheme	https
:host	siete.gursky.sk
:path	/head.png
user-agent	Mozilla/5.0...
content-type	image/png
connection	Keep-alive
cookie	id=d38rb8x39...

Posielam:

:path	/obr.jpg
content-type	image/jpg

Posielané atribúty hlavičky sú v HPACK formáte používajúcom kompresiu cez Huffmanovo kódovanie

Stav používateľovej komunikácie: cookies

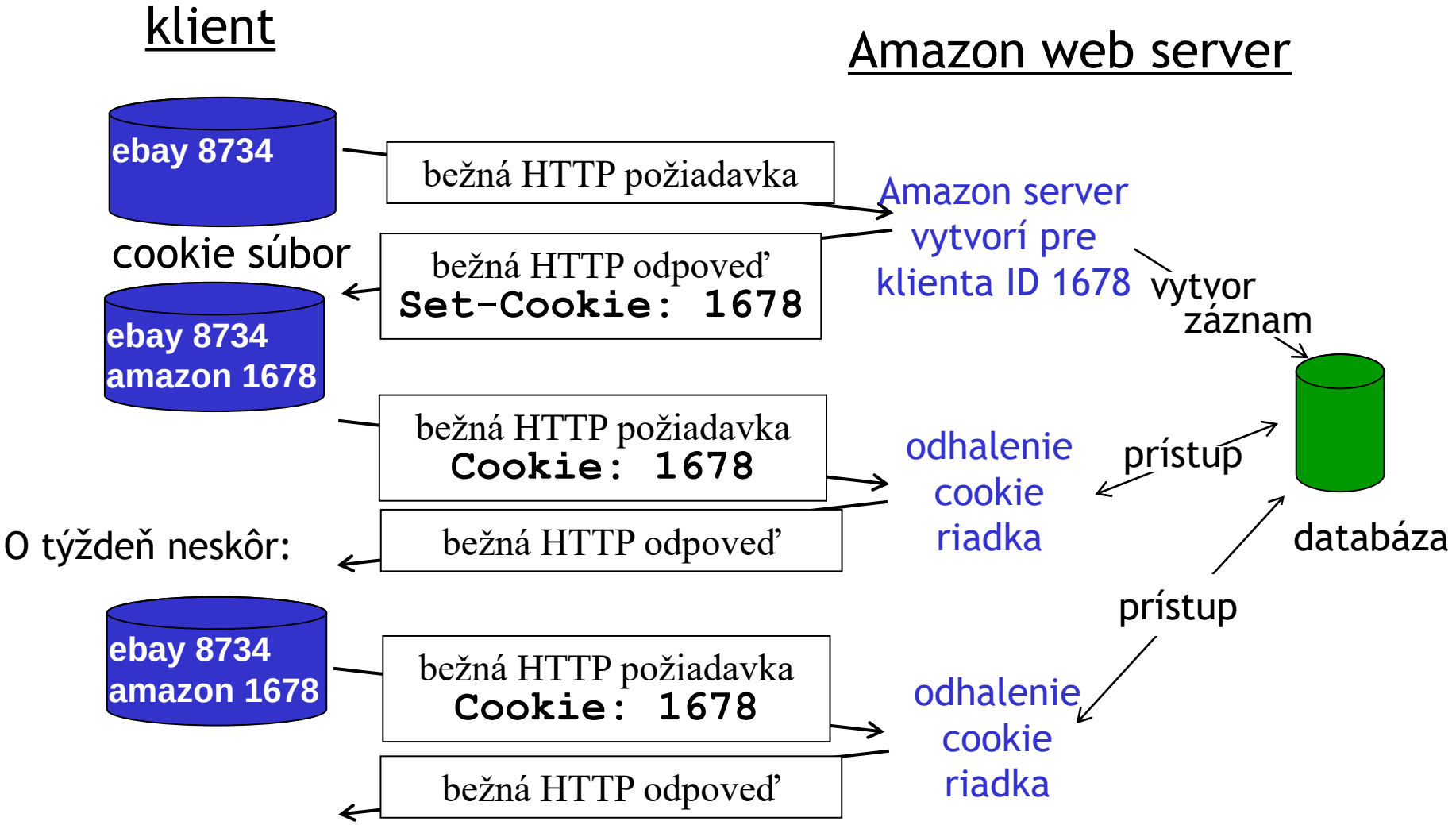
Štyri súčasti:

1) v hlavičke HTTP odpovede servera je cookie riadok
2) aj v hlavičke HTTP požiadavky môže byť cookie riadok
3) browser si uchováva cookie v špeciálnom súbore
4) web server má databázu cookie všetkých komunikácií na nejaký čas dozadu (napr. týždeň aj viac)

Postup:

- Pri prvej HTTP požiadavke server vytvorí
  - jedinečné ID
  - záznam s ID v databáze
- Toto ID sa posiela a uchováva v cookie
- Pri ďalšej návšteve sa môže browser identifikovať s ID v cookie riadku
- Webová aplikácia vie pokračovať v relácii

Stav používateľovej komunikácie: cookies



Cookies

Čo umožňujú:

- autorizáciu
- nákupné košíky
- odporúčania
- stavy komunikácie (napr. webový e-mail klient)

Cookies a súkromie:

- cookies umožňujú web serveru učiť sa správanie používateľa
- Pri nevhodne napísanej web aplikácii sa môžu do cookie správ dostať aj osobné údaje napr. z formulárov

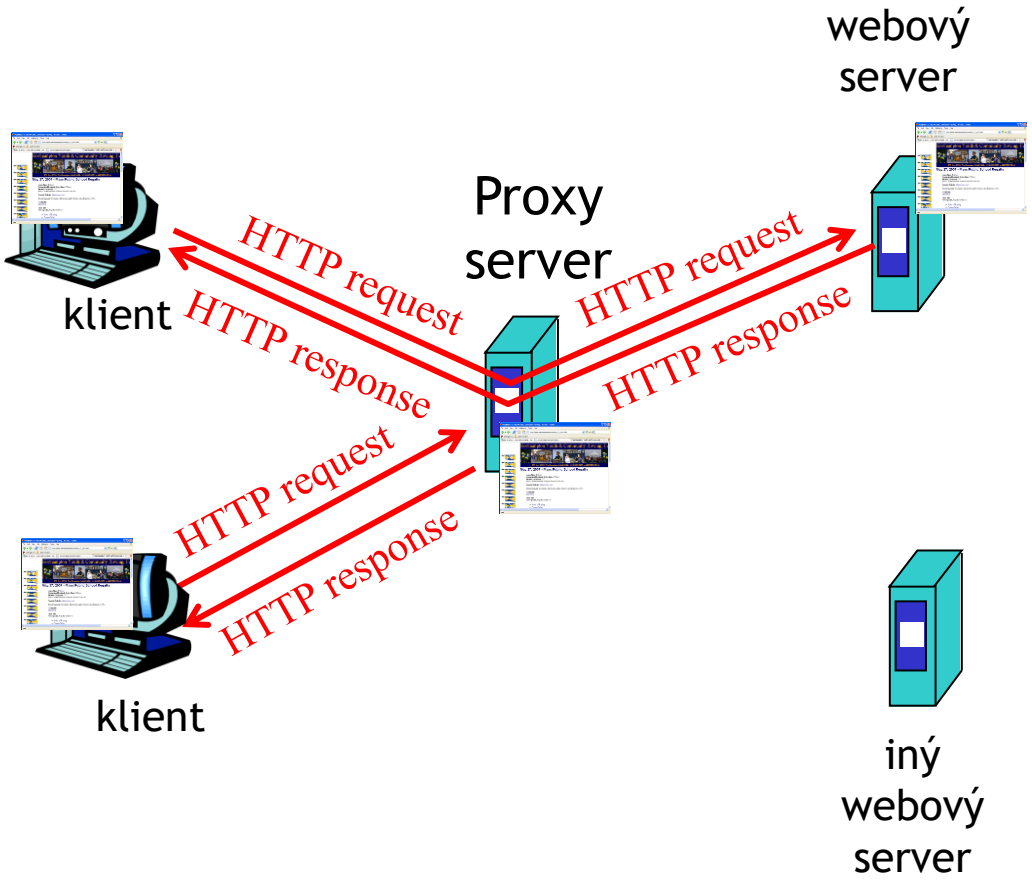
Čo udržiava "stav":

- aplikácie zabezpečujúce fungovanie dynamických webstránok si môžu ku každému cookie ID zvlášť pamätať aktuálny stav aj históriu
- cookies: stav môže byť aj súčasťou cookie v správach (spolu s ID)

Web cache (čítaj [keš]) alebo proxy server

Cieľ: obslúžiť klienta bez pričinenia pôvodného servra

- Browser môže mať nastavený prístup cez proxy server
- Browser posiela všetky HTTP požiadavky cez proxy server
  - Ak je objekt v cache proxy servera, tak sa hneď pošle klientovi
  - Inak proxy server kontaktuje pôvodný server, aby mu poslal požadovaný objekt



Proxy server

- proxy vystupuje aj ako klient, aj ako server
- bežne je proxy server riadený poskytovateľom internetu (univerzita, firma, lokálny provider)

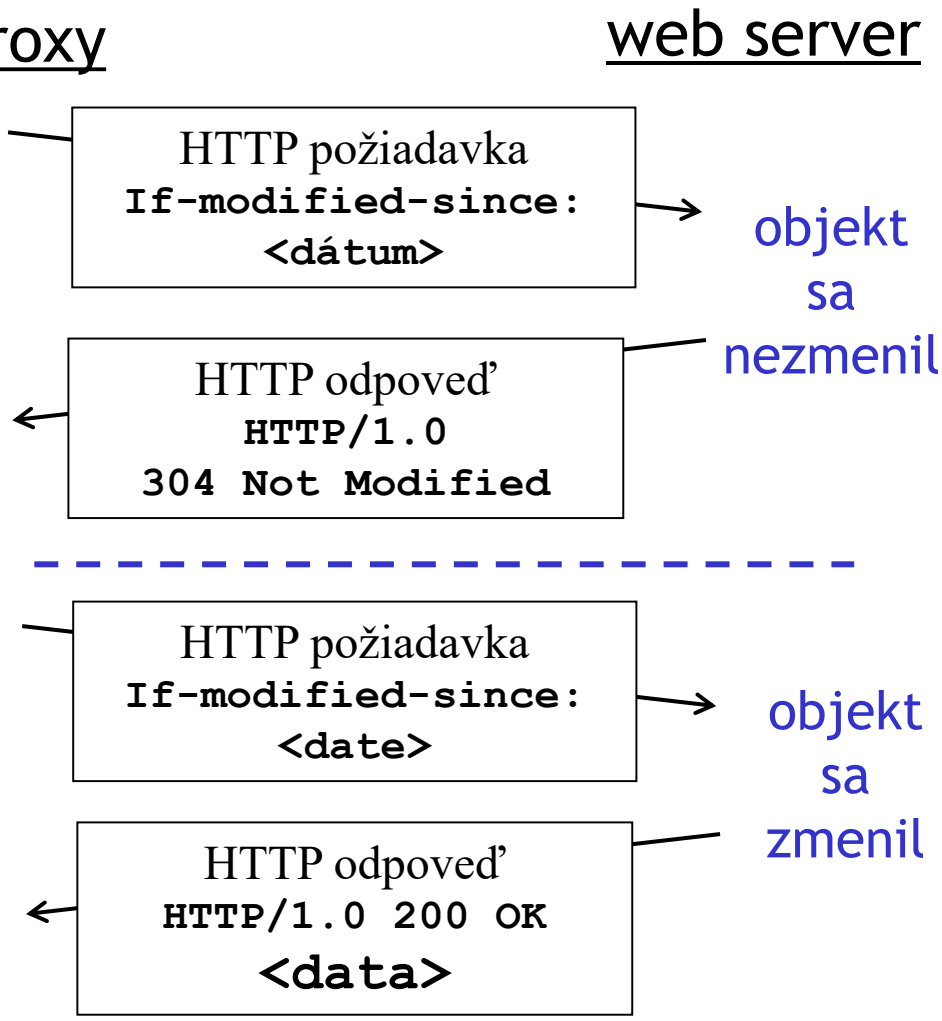
Na čo je to dobré?

- rýchlejšie zobrazenie stránky
- odľahčenie slabého sieťového pripojenia od záťaže (skôr v minulosti)
- logovanie prevádzky!
- obmedzenie prístupu na určité webstránky (pre malé deti, na školách)

Podmienený GET

- Cieľ: neposielať zbytočne z web servera objekt, ktorý je rovnaký v proxy serveri aj na zdrojovom web serveri
- Proxy dodá do HTTP požiadavky čas jeho verzie
- `If-modified-since: <dátum>`
- Server pošle buď novší objekt alebo len HTTP hlavičku odpovede s informáciou, že proxy má aktuálnu verziu:

  `HTTP/1.0 304 Not Modified`

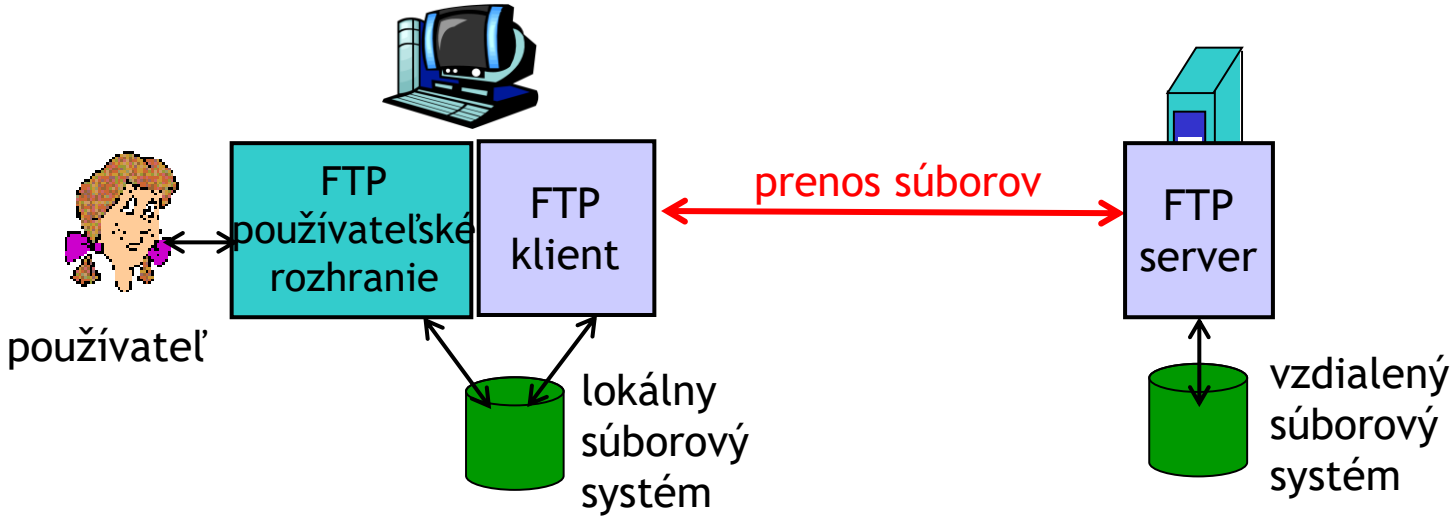


Prehľad prednášky

- úloha aplikačnej vrstvy
- využitie nižších vrstiev
- architektúry sieťových aplikácií

- aplikačné protokoly
  - HTTP
    - web
  - FTP
    - prenos súborov
  - SMTP, POP, IMAP
    - E-mail

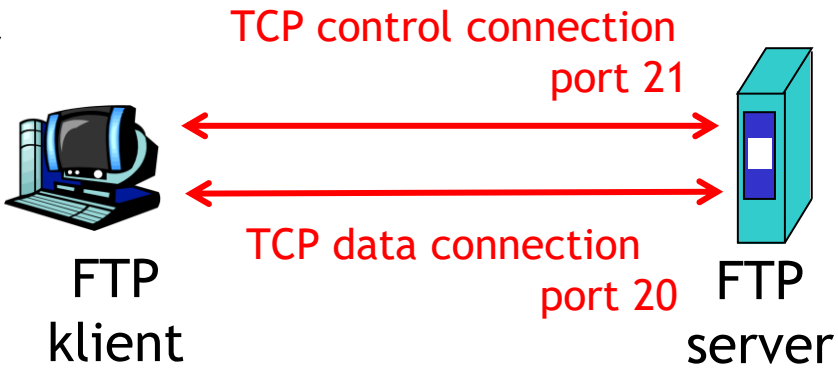
FTP: file transfer protocol



- používa sa na prenos súborov oboma smermi
- klient/server model
  - *klient:* inicializuje spojenie
  - *server:* na vzdialenom počítači
- ftp: RFC 959
- ftp server: port 21 (riadiaci – príkazy, oznamy)

FTP: oddelené riadenie a dátové prenosy

- FTP klient sa napája na FTP server na porte 21
- klient sa autorizuje cez riadiace spojenie
- klient prechádza vzdialeným súborovým systémom cez príkazy zaslané do riadiaceho spojenia
- ak príkaz vyžaduje prenos dát
  - server otvorí port pre *2. TCP spojenie* (pasívny mód)
  - klient otvorí svoj port a jeho číslo pošle servru, ktorý inicializuje toto spojenie z portu 20 (aktívny mód)



- po odoslaní 1 súboru server ukončí dátové spojenie (riadiacim kanálom dôjde správa o úspešnom kopírovaní)
- na poslanie ďalšieho súboru sa musí otvoriť ďalšie dátové spojenie
- FTP server udržiava “stav” klienta: aktívny adresár, prihlásenie

FTP príkazy a odpovede

Niektoré príkazy:

- **USER *username***
- **PASS *password***
- **LIST** – vráti obsah aktuálneho adresára (cez dátové spojenie)
- **RETR filename** žiada súbor zo servra
- **STOR filename** posiela súbor na server

Príklady návratových správ:

- Kód a komentár stavu (ako pri HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

Prehľad prednášky

- úloha aplikačnej vrstvy
- využitie nižších vrstiev
- architektúry sieťových aplikácií

- aplikačné protokoly
  - HTTP
    - web
  - FTP
    - prenos súborov
  - SMTP, POP, IMAP
    - E-mail

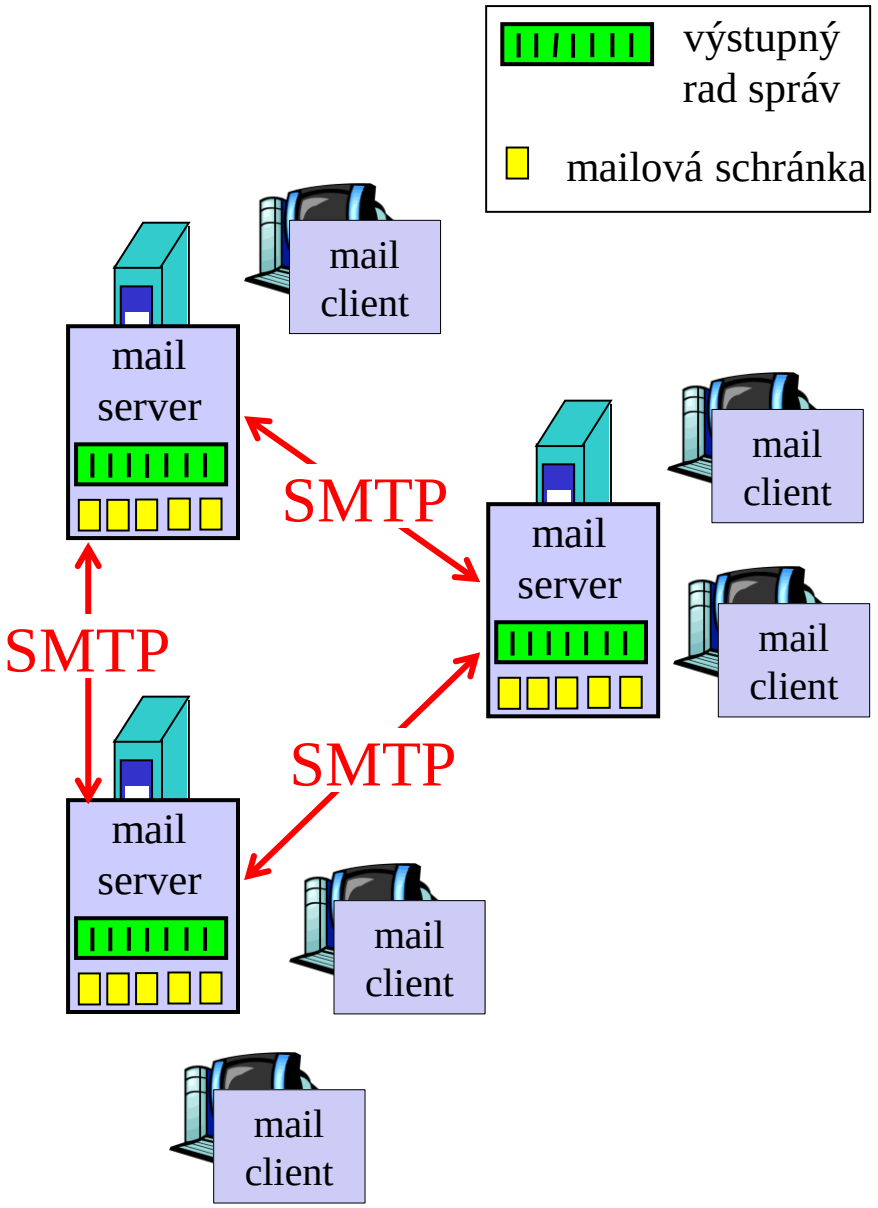
E-Mail

Tri hlavné komponenty:

- mailoví klienti (user agents)
- mailové servery
- simple mail transfer protocol: SMTP

Mailový klient

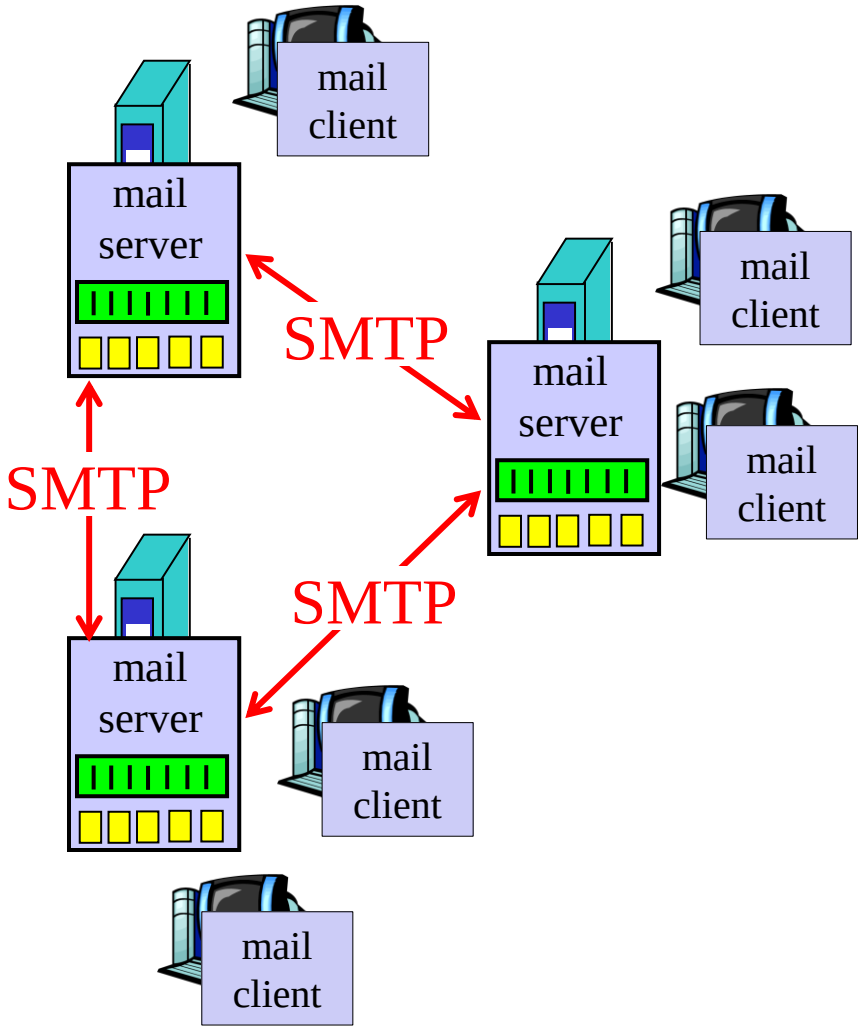
- Posielanie, editovanie a čítanie mailov
- napr. Outlook, elm, Mozilla Thunderbird, Evolution
- odchádzajúce a prichádzajúce maily sú uložené na serveri



Mailový server

obsahuje:

- Mailové schránky každá obsahuje doručené maily pre používateľa
- Rad mailov obsahujúci maily určené na odoslanie
- SMTP protokol na komunikáciu a posielanie mailov medzi mailovými servermi
  - "klient": mailový server odosielajúci mail
  - "server": mailový server prijímajúci mail

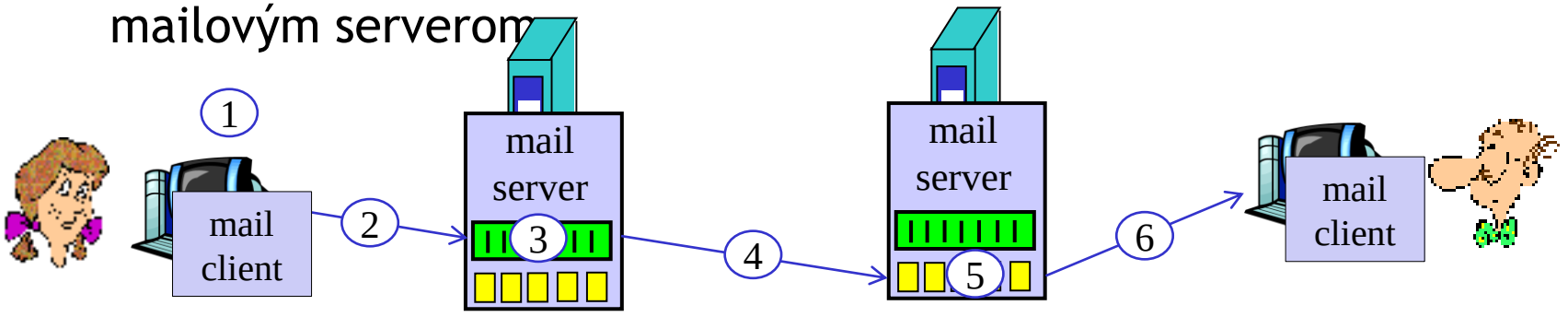


E-mail: SMTP [RFC 2821]

- používa TCP protokol na spoľahlivé doručenie mailov z klienta na server počúvajúci na porte 25
- tri fázy prenosu z odosielajúceho servera na prijímací
  - pozdravenie
  - prenos správ
  - ukončenie
- komunikácia cez príkazy a odpovede
  - príkazy: ASCII text
  - odpovede: kód a komentár stavu
- Správy musia byť v 7-bitovom ASCII kódovaní

Scenár: Alica posiela mail Bobovi

1) Alica použije svojho mailového klienta na odoslanie mailu pre `bob@inštitúcia.sk`
2) Alicin mailový klient pošle mail jej mailovému serveru, ktorý uloží mail do radu mailov
3) Alicin mailový server otvorí TCP spojenie s Bobovým mailovým serverom
4) SMTP klient (Alicin mailový server) pošle Alicin mail cez TCP spojenie
5) Bobov mailový server uloží mail do Bobovej schránky
6) Bobov mailový klient si po čase vypýta mail z jeho schránky na jeho mailovom serveri



Príklad SMTP komunikácie

```
S: 220 kosice.upjs.sk
C: HELO doma.sk
S: 250  Hello doma.sk, pleased to meet you
C: MAIL FROM: <alica@doma.sk>
S: 250 alica@doma.sk... Sender ok
C: RCPT TO: <bob@upjs.sk>
S: 250 bob@upjs.sk ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Prepacte, ze som pred tyzdnom neposlala projekt
C: lebo som velmi chora
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 kosice.upjs.sk closing connection
```

SMTP: na záver

- SMTP používa stále spojenia
- SMTP vyžaduje, aby celá správa (hlavička aj telo) boli v 7-bitovom ASCII kódovaní
- Koniec správy označuje postupnosť `CRLF.CRLF`

Porovnanie s HTTP:

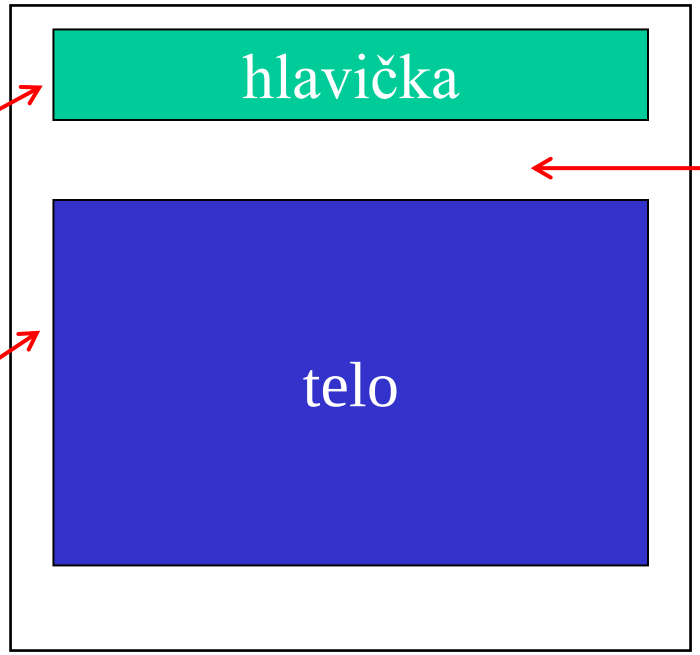
- HTTP: pull (ťahanie)
- SMTP: push (tlačenie)
- Oba používajú ASCII kódovanie pre príkazy a hlavičky odpovedí
- HTTP: každý objekt na stránke je nezávislý a ťahaný nezávisle
- SMTP: všetko vrátane príloh je posielané v jednej správe

Formát mailov

SMTP: protokol na výmenu mailových správ

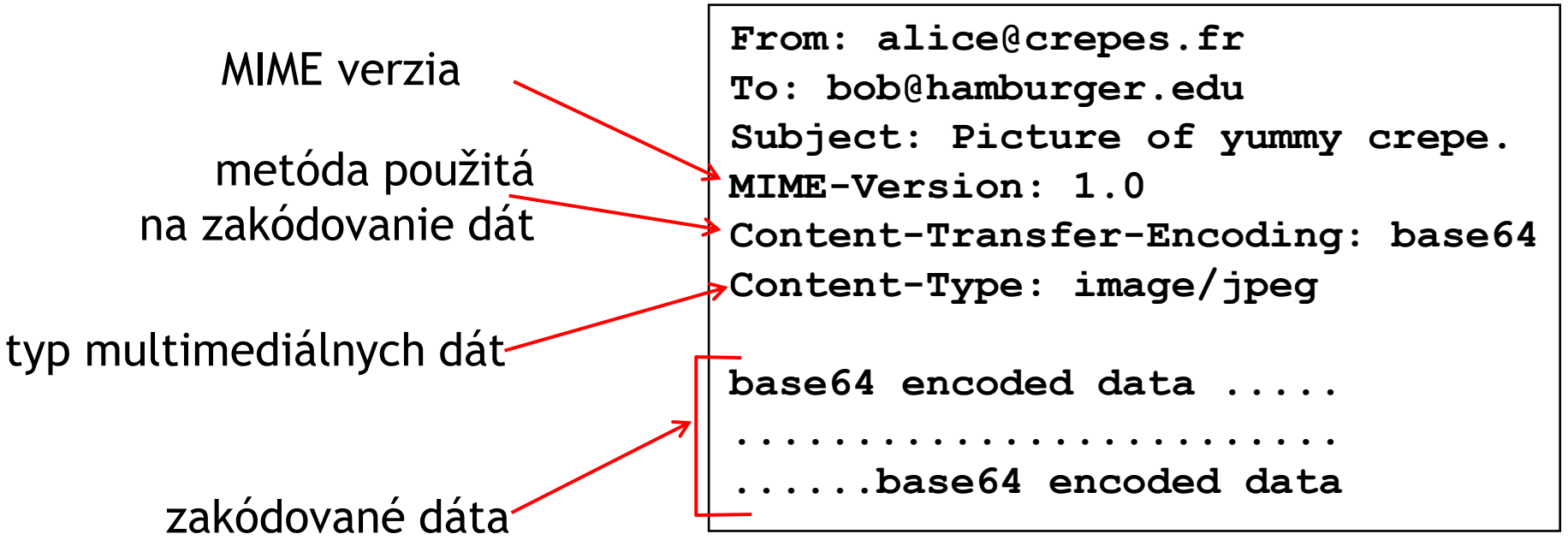
RFC 822: štandard pre textové správy:

- Riadky v hlavičke napr.
  - To:
  - From:
  - Subject:
- telo
  - Samotná správa
  - Musí byť v 7-bitovom ASCII kódovaní

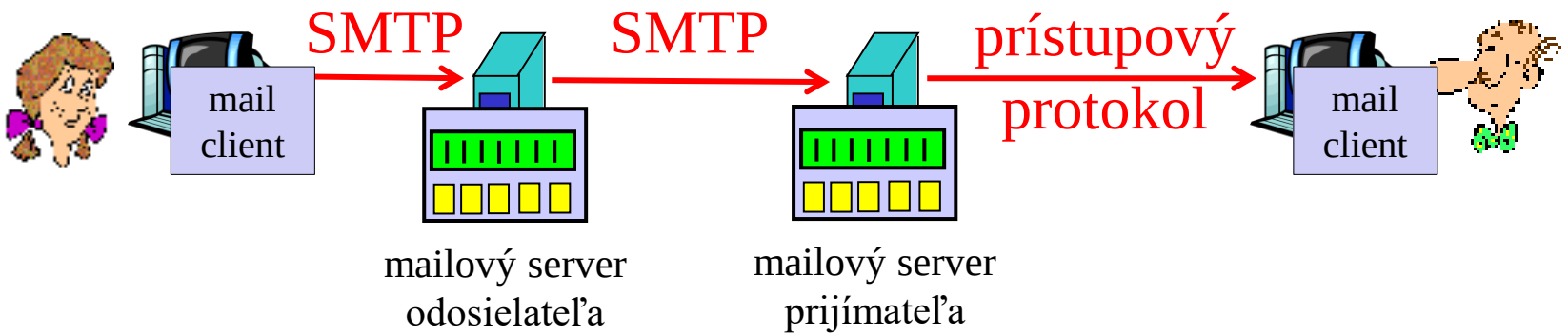


Formát mailov: multimediálne rozšírenia

- MIME: multimedia mail extension, RFC 2045, 2056
- dodatočné riadky v hlavičke označujú MIME typ



Protokoly na prístup k mailom



- SMTP: na doručenie na mailový server prijímateľa (adresáta)
- Protokol na prístup k mailom: získanie mailu zo servera
  - POP: Post Office Protocol [RFC 1939]
    - autorizácia a stiahnutie
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - viac možností
    - Správa uložených mailov priamo v schránke na servri
  - HTTP: gmail, Hotmail, Yahoo! Mail, Zoznam, Post, Azet, ...

POP3 protokol

Autorizačná fáza

- Klientské príkazy:
  - **`user`**: používateľské meno
  - **`pass`**: heslo
- Odpovede servera
  - **`+OK`**
  - **`-ERR`**

Fáza transakcií

- **`list`**: zoznam čísiel mailov
- **`retr`**: získanie mailu s daným číslom
- **`dele`**: vymazanie
- **`quit`**: koniec spojenia

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass tajne
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <obsah mailu 1>
S: .
C: dele 1
C: retr 2
S: <obsah mailu 2>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 a IMAP

Viac o POP3

- V príklade sa použil mód “stiahni a vymaž”
- Ak používateľ sadne k inému počítaču, k starým mailom sa nedostane
- Bez mazania máme kópie tých istých mailov na viacerých strojoch

IMAP

- Necháva maily na serveri
- Umožňuje organizáciu mailov do priečinkov na serveri

Ďakujem za pozornosť

Modifikované slajdy z knihy:

Computer Networking: A Top Down Approach ,
4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.