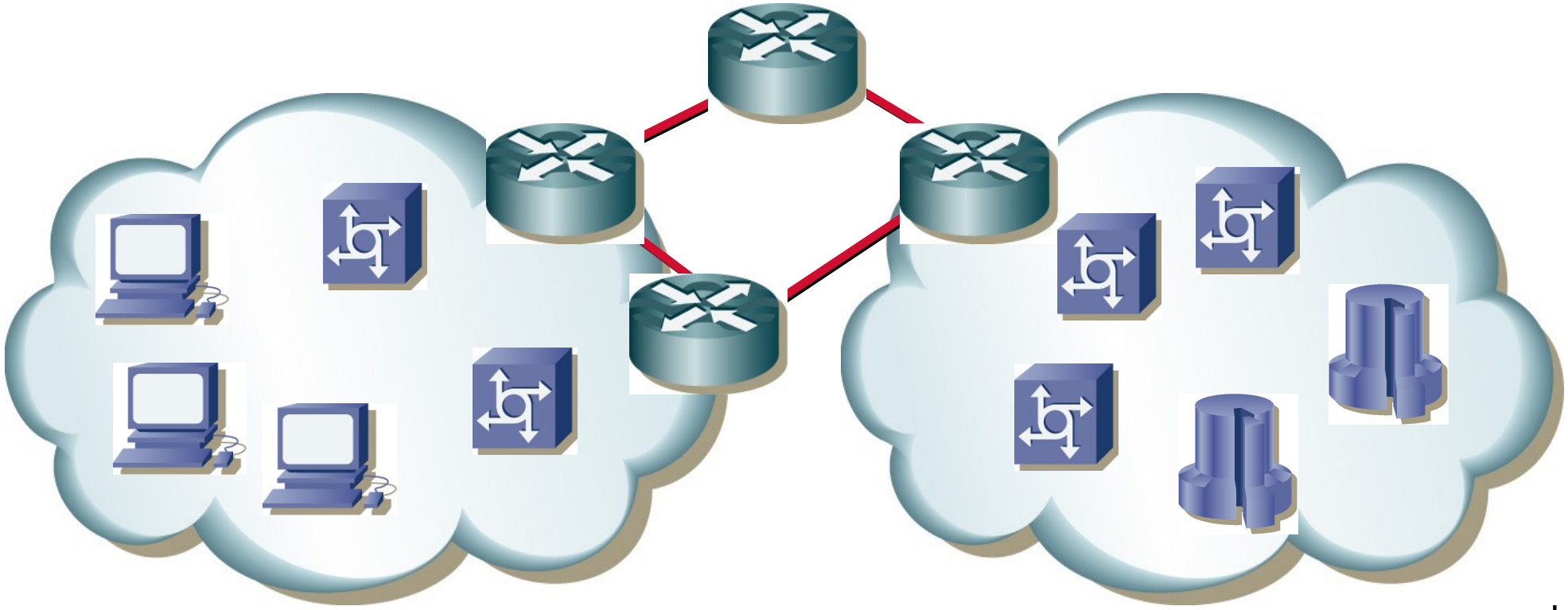
Počítačové sítě II

18. World Wide Web, HTTP

Miroslav Spousta, 2006
<qiq@ucw.cz>, http://www.ucw.cz/~qiq/vsfs/



Historie WWW

- World Wide Web
- v současnosti nejrozšířenější a nejpoužívanější služba Internetu
  - nebylo tomu tak vždy (Gopher, ...)
- vyvinut v roce 1989 v Cernu ve Švýcarsku (Tim Berners-Lee)
  - pro potřebu sdílení informace fyziků
  - původně jako textová služba
  - základní idea: hypertext
- 1992 – 1993: vývoj prohlížeče NSCA Mosaic
  - volně šiřitelný, dostupný
- 1994: založena firma Netscape Communications
  - vznik prohlížeče Netscape Navigator
- obsah sdělení (HTML, XHTML), přenosový protokol (HTTP)

HTTP

- HyperText Transport Protocol, RFC 2616 (HTTP 1.1)
- jednoduchý textový protokol
- používá spolehlivou službu: TCP, port 80
- funguje bezestavově
  - nemění se stav serveru
  - pro každý objekt se navazuje nové spojení (HTTP verze 1.0)
- architektura klient – server
  - klient naváže spojení se serverem
  - pošle svůj požadavek
  - server odpoví, pošle výsledek
  - dojde k uzavření spojení (HTTP verze 1.0), případně následuje další požadavek (HTTP verze 1.1)
- umožňuje cachování odpovědí

HTTP metody

- příkazům klienta se říká metody
- pro různé akce existují různé metody
  - GET, HEAD, POST, PUT, DELETE, OPTIONS, TRACE
- příkaz klienta vypadá takto:
  - specifikace metody, objektu a verze protokolu
  - možné hlavičky požadavku
  - prázdný řádek (CRLF)
  - případně tělo (např. u PUT)

```
GET /index.html HTTP/1.0
Host: www.seznam.cz
User-Agent: Mozilla/5.0
Referer: http://google.com?q=xxx
```

- absolutní cesta v URI požadavku (včetně hostname) může být pouze při použití proxy serveru, používá se hlavička Host:

HTTP odpověď

- server vyřídí požadavek a vrátí odpověď
- první řádek udává verzi protokolu, chybový kód a jeho textovou reprezentaci
- následují hlavičky serveru, prázdný řádek a tělo odpovědi
- chybové kódy jsou podobné jako u SMTP:
- 2xx: bez chyby
  - 200 OK, 201 Created, 202 Accepted, ...
- 3xx: přesměrování klienta
  - 301 Moved Permanently, 302 Found, 304 Not modified
- 4xx: chyba u klienta
  - 400 Bad request, 403 Forbidden, 404 Not found
- 5xx: chyba na serveru
  - 500 Internal server error, 501 Not implemented

```
C: GET /index.html HTTP/1.0
C: Host: www.seznam.cz
C:
S: HTTP/1.0 404 Not found
S: ...
```

Používané metody

- metoda GET
  - umožňuje získat stránku (resource) ze serveru (může to být dynamický obsah)
  - je to nejčastěji používaná metoda
  - následuje cesta a verze protokolu HTTP
- metoda HEAD
  - používá se stejně jako GET, ale nevrací obsah (stránku)
  - slouží k testování, např. zda byla stránka změněna (zjistí se z hlavičky odpovědi)
- metoda POST
  - umožňuje předat data serveru
  - např. pro odeslání dat z formuláře
- metoda PUT
  - jako POST, ale měla by se použít, pokud dojde k náhradě entity obsahem

POST vs GET

- při použití dynamických stránek (CGI, php, ASP, …) je možné předávat parametry serveru (např. ve formuláři) pomocí metod POST a GET
  - v metodě GET se předávají jako součást URL (jako parametr za otazníkem)
    - GET /search?hl=en&q=test HTTP/1.1
  - v metodě POST se předávají v těle požadavku
- liší se hlavně v sémantice:
  - GET by se mělo používat, pouze pro získání dané stránky s určitými parametry, server by měl pro dané argumenty vracet stejnou stránku
  - POST lze použít i pro uložení dat, odeslání e-mailu, atd.
- problém při použití POST: vypršení stránky (expire)
  - má dobrý důvod: většinou nechcete objednat zboží 2x
  - řešení: stránka, na kterou míří POST provede zpracování údajů a poté ihned přesměrování na cílovou stránku pomocí kódu 303 (302 pro starší klienty); stránka, s kódem 303 se NESMÍ cachovat

Málo používané metody

- metoda DELETE
  - smaže obsah URL
- metoda OPTIONS
  - získáme použitelné metody pro dané URL
  - příliš se nepoužívá
- metoda CONNECT
  - pro otevření přístupu na proxy server
  - nutnost při použití proxy serveru a SSL spojení
- metoda TRACE
  - vrátí zpět to, co dostala na vstupu
  - pro debugování

Hlavičky klienta

- Date:
  - čas a datum vytvoření zprávy (ve formátu RFC 822)
- Referer:
  - odkaz na URL, ze kterého požadavek pochází
  - umožňuje zjišťovat, odkud uživatelé na stránku přicházejí
- User-Agent:
  - verze klienta (např. User-Agent: Mozilla/5.0)
  - pro statistické účely
- Content-Type, Content-Length, Transfer-Encoding
  - jako v SMTP poště (MIME), velikost dat
- From:
  - e-mailová adresa člověka zodpovědného za dané dotazy
  - typicky se používají pro roboty – aby bylo jasné, komu nadávat

Hlavičky klienta (2)

- Accept:
  - které typy odpovědi klient akceptuje (text/html, text/plain, */*)
  - umožňuje použít i preference, vyjadřují se jako číslo menší než 1
  - např. text/plain; q=0.2, text/html; q=0.88, */*; q=0.1
- Accept-charset:, Accept-encoding:
  - podobně jako u Accept umožňuje použít preference
  - charset: např. iso-8859-1, unicode
  - encoding: např. gzip, compress, identity
- Accept-language:
  - preferované jazyky dokumentu, opět jsou možné preference
  - např. cz; q=0.8, en-gb; q=0.7, en; q=0.5

Hlavičky serveru

- Allow:
  - pokud server vrátil 405 Method not allowed, říká, které metody jsou pro URL povolené
- Date:
  - kdy byla odpověď odeslána
- Location:
  - odkaz, kam vede přesměrování (kódy 301, 302 nebo 303)
- Server:
  - verze SW, podobně jako User-Agent
- Content-Length, Content-Type, Transfer-Encoding
  - stejné jako u klienta

Hlavičky (cachování)

- Pragma: no-cache
  - řídí cachování v HTTP 1.0
- Cache-Control:
  - v HTTP 1.1, umožňuje nastavovat cachování podrobně
  - max-age, public, private, no-cache
- Expires:
  - kdy stránka má propadnout v cache
- If-(Un)Modified-Since:
  - následuje datum (v GMT)
  - podmíněný požadavek, vrací stránku pouze pokud byla změněna od uvedeného data
  - jinak vrátí 304 Not modified
- Etag: identifikace instance daného objektu, If-(None-)Match

Basic authentication

- chceme omezit přístup ke stránkám na některé uživatele
  - ověřovat jménem a heslem
- HTTP poskytuje možnost: basic authentication
- při přístupu na zabezpečenou zprávu vygeneruje server odpověď
  - 401 Unauthorized
  - do hlavičky odpovědi přidá položku WWW-Authenticate: Basic realm="xxx"
- klient se zeptá uživatele na jméno a heslo, vytvoří řetězec „jmeno:heslo"
- ten se pomocí uuencode algoritmu převede na řetězec
- klient pošle v hlavičce požadavku Authorization: Basic aWFuOmJvb2J5
- realm slouží k tomu, aby klient rozlišil, které heslo má poslat

Cookies

- původní návrh HTTP protokolu je bezestavový
  - což je velké plus, kvůli jednoduché implementaci
  - ale může způsobit problémy při psaní webových aplikací
  - občas potřebujeme mít souvislost mezi jednotlivými přístupy uživatelů
- řešení: cookies
- server současně s odpovědí vrátí v hlavičce položku Set-Cookie:
  - obsahem položky je řetězec
  - může specifikovat i další atributy (např. životnost)
  - klient si cookie uloží do souboru, pamatuje si k ní server, ze kterého přišla
- klient při odesílání požadavku vždy prozkoumá svojí databázi cookies
- najde-li cookie, která odpovídá serveru, pošle ji spolu s požadavkem
  - v hlavičce jako Cookie:

HTTP 1.1

- HTTP 1.0 pro každý požadavek navazuje nové spojení
  - např. pro každý obrázek na stránce
  - neefektivní, zbytečná zátěž serveru i klienta, sítě
  - důsledek bezestavovosti
- HTTP 1.1 dovoluje požadavky sdružit do jednoho spojení
  - pomocí jednoho spojení je možné stáhnout více objektů z jednoho serveru
  - efektivnější, rychlejší
  - podpora komprimace dat (gzip, compress)
  - podpora částečných přenosů
  - povinná hlavička Host:
  - umožňuje vytvářet více virtuálních serverů s jednou IP adresou
  - umožňuje posílat data postupně (stream): chunked encoding

Hlavičky (HTTP 1.1)

- Cache-control:
  - nahrazuje HTTP 1.0 hlavičku Pragma: no-cache
- Connection: close
  - signalizuje protistraně, že spojení bude uzavřeno po skončení přenosu požadavku
- Range:
  - chceme jen část obsahu, specifikujeme rozsah bajtů, které chceme
- Content-MD5
  - pro zjištění, zda přenesená data jsou v pořádku
- Host:
  - povinný (v HTTP 1.0 volitelný)
  - udává, který server má být požadavkem osloven, může obsahovat port
  - Host: www.idnes.cz:80

Virtuální servery

- chceme na jednom fyzickém serveru provozovat více webových serverů
  - s různým DNS jménem
  - pro různé účely, případně zákazníky
- v zásadě existují dvě možnosti: name based a IP based
- IP based virtuální servery
  - každý virtuální server má svoji IP adresu, která je přiřazena serveru
  - na portu 80 pro každou z adres běží jiná instance web serveru
- name based virtuální servery
  - všechny serveru sdílí jednu IP adresu
  - rozlišení se děje na základě položky hlavičky požadavku Host: (povinná pro HTTP 1.1)
  - někteří staří klienti nemusí podporovat (dnes už podporují všichni)
  - problém s SSL komunikací

Chunked encoding

```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-Type: text/plain
Transfer-Encoding: chunked
1a; ignore-stuff-here
abcdefghijklmnopqrstuvwxyz
10
1234567890abcdef
0
some-footer: some-value
another-footer: another-value
[blank line here]
```