Katedra softwarového inženýrství,
Matematicko-fyzikální fakulta,
Univerzita Karlova, Praha

Rodina protokolů TCP/IP,
verze 2.7

Část 10: World Wide Web

Jiří Peterka, 2011

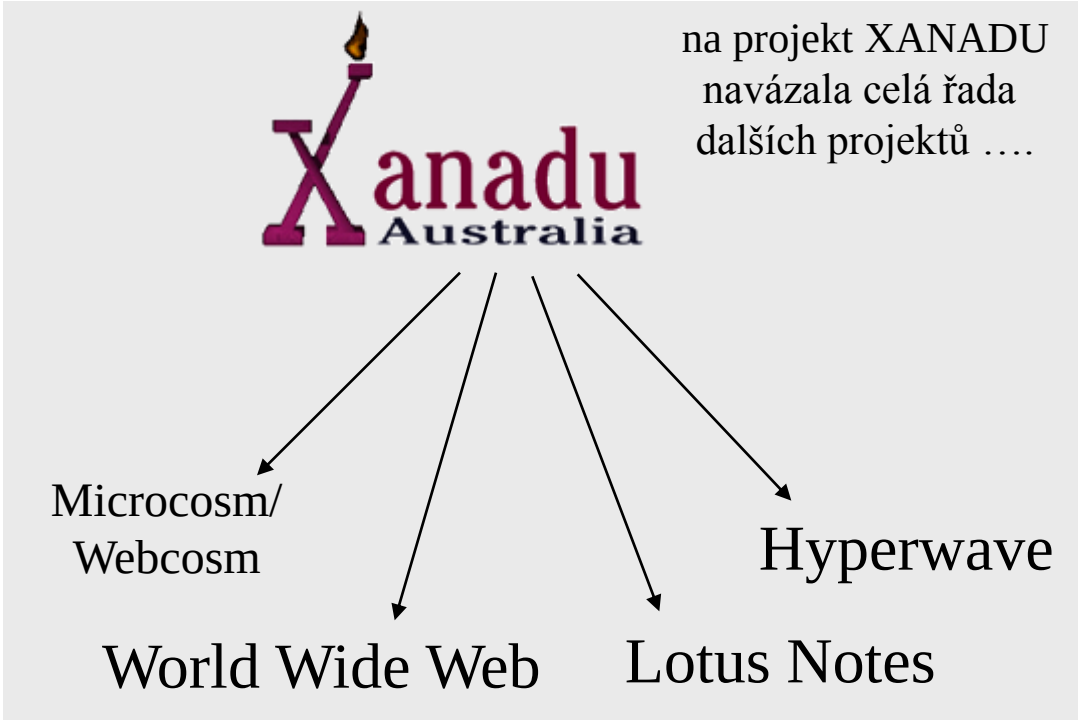
Prehistorie WWW

- dr. Vannevar Bush
  - za 2. světové války řídil práci cca 6000 amerických vědců
    - směrovanou k aplikacím vědy do vojenství (zbraní)
  - po válce se snažil motivovat vědce aby zaměřili své úsilí "civilním směrem"
    - teze: lidé se až dosud zabývali především posílením svých "hmotných" schopností
      - konstruováním strojů, nástrojů, …
    - nyní by se měli zaměřit na posilování svých intelektuálních schopností, na zužitkování vědomostního potenciálu, který se za celou existenci lidstva shromáždil
  - vyšel z představy, že lidský mozek používá "asociace", prostřednictvím kterých propojuje jednotlivé vzpomínky a znalosti (data) a vytváří celou "pavučinu"
    - chtěl přenést tento koncept i do "fyzického světa"
  - navrhl zařízení nazvané MEMEX, určené k organizování informací na principu asociací
    - s možností propojovat jednotlivé části informací pomocí "spojů", které mohou být i vícenásobné a které lze anotovat (popisovat)
  - svou koncepci publikoval v revolučním článku **AS WE MAY THINK**
    - vyšlo v červenci 1945, v časopise The Atlantic Monthly
    - http://www.csi.uottawa.ca/~dduchier/misc/vbush/as-we-may-think.html

Prehistorie WWW

- Vannevar Bush je považován za otce myšlenky (koncepce) dnešního hypertextu
  - sám termín "hypertext" nepoužil
  - jeho MEMEX nebyl nikdy zkonstruován

- autorem termínu "hypertext" a "hypermedia" je Ted (Theodore) Nelson
  - navrhl systém sdílení informací jménem XANADU
    - na principu "pavučiny" (nelineárního uspořádání s asociacemi)
    - je obecnější než dnešní WWW
      - pracuje s více formáty, má oboustranné odkazy, zahrnuje "transclusions" (obdoba include), počítá s placením za přístup ….
    - systém XANADU byl implementován!!!!
      - ale nebyl nikdy prodáván jako produkt, byl jen vývojovým prototypem !!!!



historie WWW

- **1989:** vyvinut ve středisku CERN (Švýcarsko)
  - jako prostředek sdílení informací v komunitě fyziků vysokých energií (*Tim Berners-Lee*) - především jako textová služba
- **1992** (*Andreesen, Bina*):
  - rozhodnutí vyvinout NCSA Mosaic
- **1993:** NCSA Mosaic k dispozici a volně šiřitelný

  ..... WWW začíná získávat velkou popularitu
- **1994** (*Clark, Andreesen*): založení Netscape Communications, browser Navigator

  ... WWW se začíná prosazovat i v „komerčním světě“

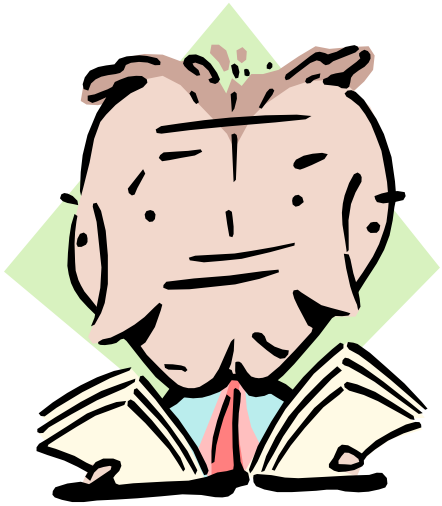
  … vzniká W3C consorcium
- **1995**: Microsoft si všímá Internetu …
  - browser Internet Explorer (využívá kód v licenci od NCSA)
- **léto 1996:**
  - začíná "válka browserů" (MS IE 3.0 vs. NS Navigator)

Tim Berners-Lee

Marc Andreesen

Bill Gates

co je hypertext?



Pozorování: lidé nemyslí přímočaře!

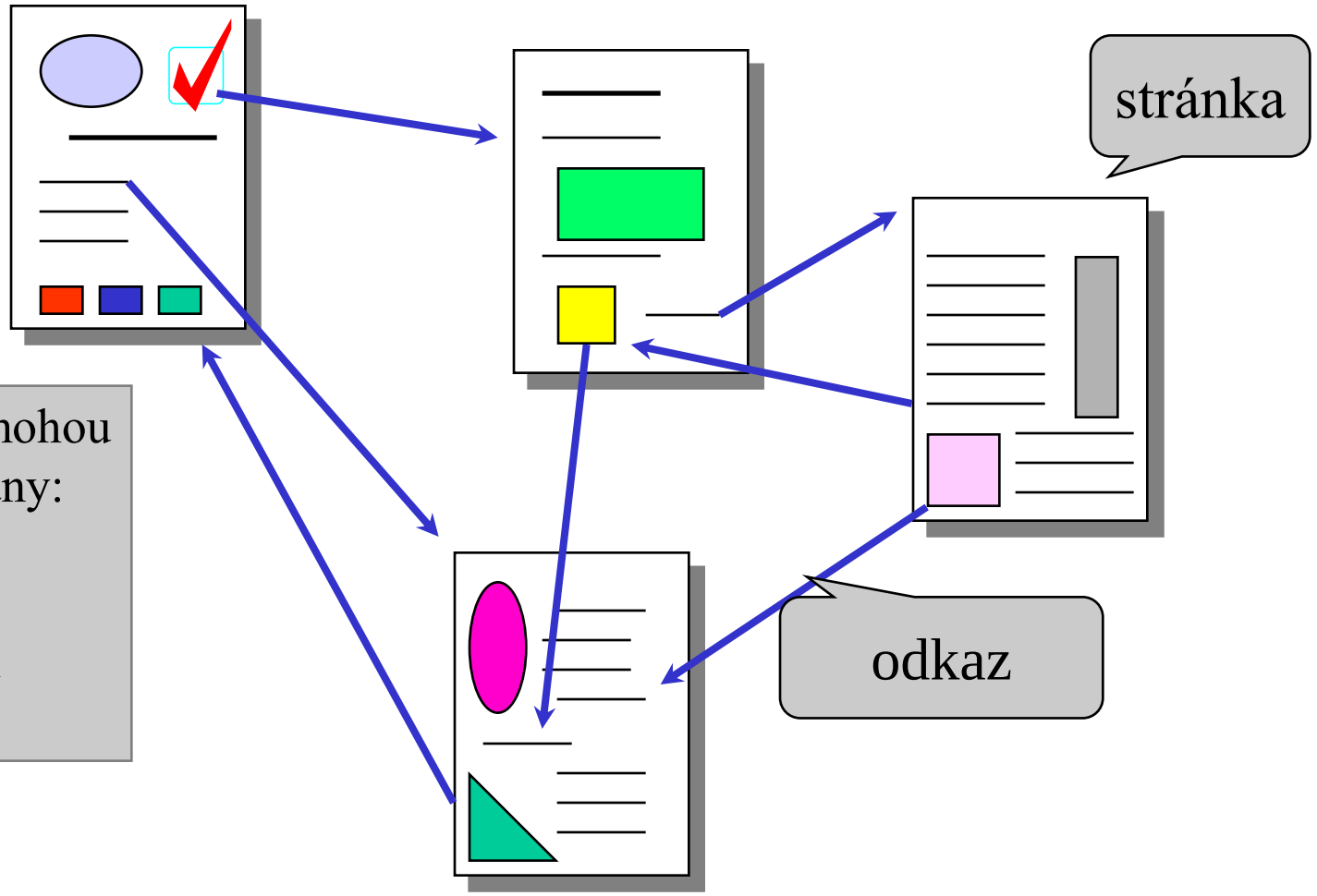
..... ale často skáčí z myšlenky na myšlenku, na základě asociací ...

- **hypertext** je (původně lineární) text, uzpůsobený pro přeskakování „z myšlenky na myšlenku“
- přeskakování v hypertextu je označováno jako brouzdání (browsing, to browse)

informace v hypertextové podobě

- jsou členěny na (relativně malé) jednotky, zvané **stránky** (pages)
  - v rámci stránek jsou informace uspořádány (víceméně) lineárně
- stránky se mohou nalézat na různých (libovolných) místech
  - je to plně distribuované
- mezi stránkami mohou existovat libovolné **vazby**
  - stránky mohou být vzájemně provázány dle libosti
  - vazby jsou **aktivní odkazy**, jejich *navolením* lze vyvolat přechod na jiné místo
- práce s hypertextem je **brouzdání** (browsing)
  - postupné procházení stránkami, s přeskoky pomocí odkazů
  - uživatel (čtenář) si sám volí jak bude stránkami procházet
    - podle svých zájmů, potřeb, schopností
    - "přijímá informace" v takovém sledu, objemu a rychlosti, jaký mu vyhovuje
- hypertextová podoba je "autorským dílem"
  - záleží na autorovi, jak dobře či špatně (přehledně či nepřehledně) rozdělí původně "lineární" informace do jednotlivých stránek a jak je mezi sebou prováže
    - jak dokáže anticipovat potřeby čtenářů

představa hypertextu (ve WWW)



aktivní odkazy mohou být reprezentovány:
- částmi textu
- obrázky
- částmi obrázků
- ……

nabídka (menu) je jakoby vnořena přímo do obsahu (textů, obrázků atd.)

odbočení: Gopher

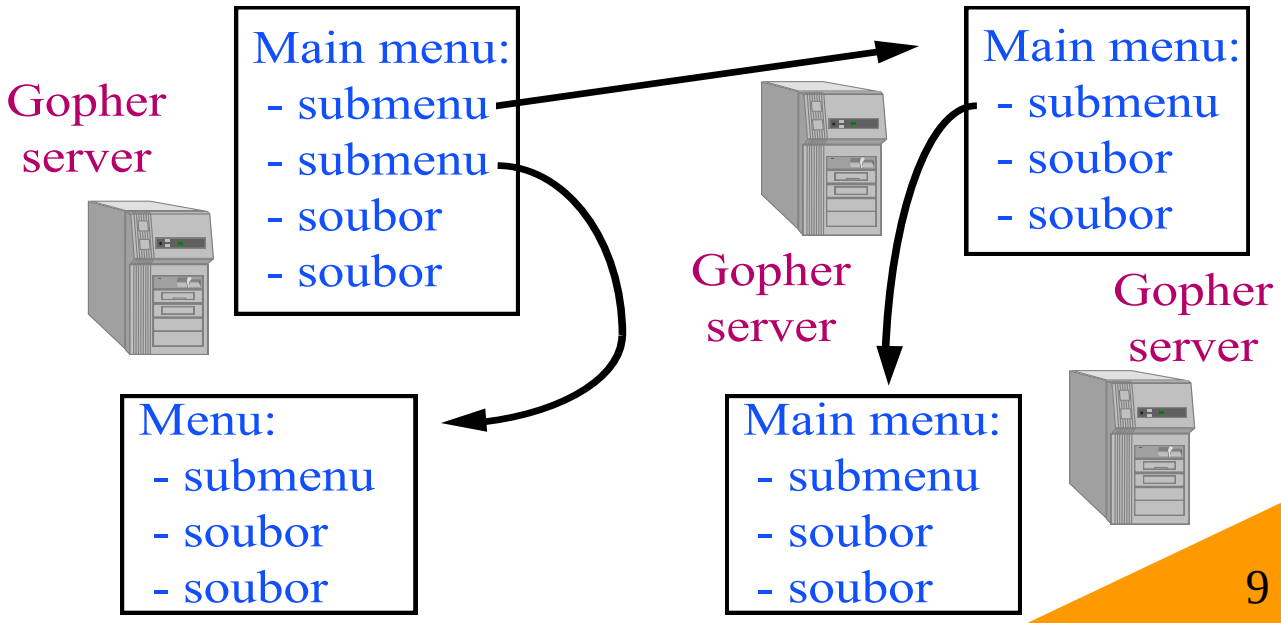
- Gopher měl stejné ambice jako World Wide Web
  - ale nebyl (tolik) postaven na hypertextu, jako World Wide Web!!!

- Gopher byl vyvinut na University of Minnesota, USA
  - jako služba pro zpřístupnění informací
- uživateli poskytuje nabídku ve formě menu
  - jednotlivé položky menu jsou uspořádány lineárně
  - položky jsou textové (i celé menu)
- položka může představovat odkaz na:
  - soubor (text, obrázek, ...)
  - odkaz na jiné menu
  - přechod (bránu) do jiné služby či aplikace
- vlastní obsah (text. obrázky) je oddělen od nabídky (menu) !!!

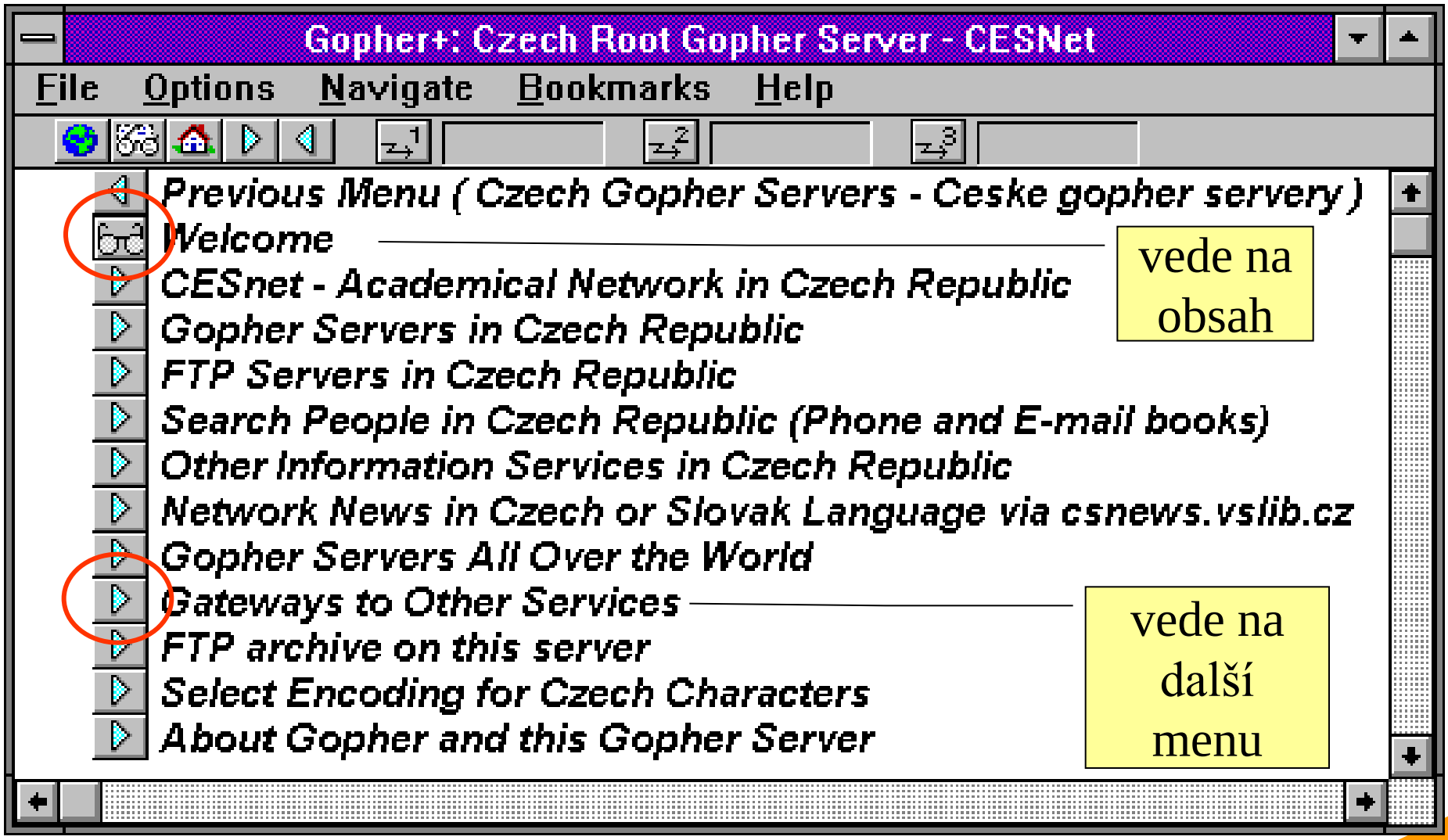
Gopher a celosvětová pavučina

- položky v menu služby Gopher nemusí reprezentovat pouze místní zdroje (soubory, další menu, ...)
  - mohou to být odkazy na zdroje nacházející se na jiných uzlech, po celém Internetu
    - jednotlivé Gopher servery se mohou odkazovat na sebe navzájem a na své zdroje
    - vzájemné „provázání“ prostřednictvím odkazů není nijak apriorně omezeno
    - mohou existovat i „prázdné“ servery, které nenabízí žádné vlastní zdroje

- díky vzájemnému provázání menu služby Gopher vzniká skutečná celosvětová pavučina!!
  - **ale v "souboji o celosvětovou pavučinu" zvítězil hypertextový WWW !!!**

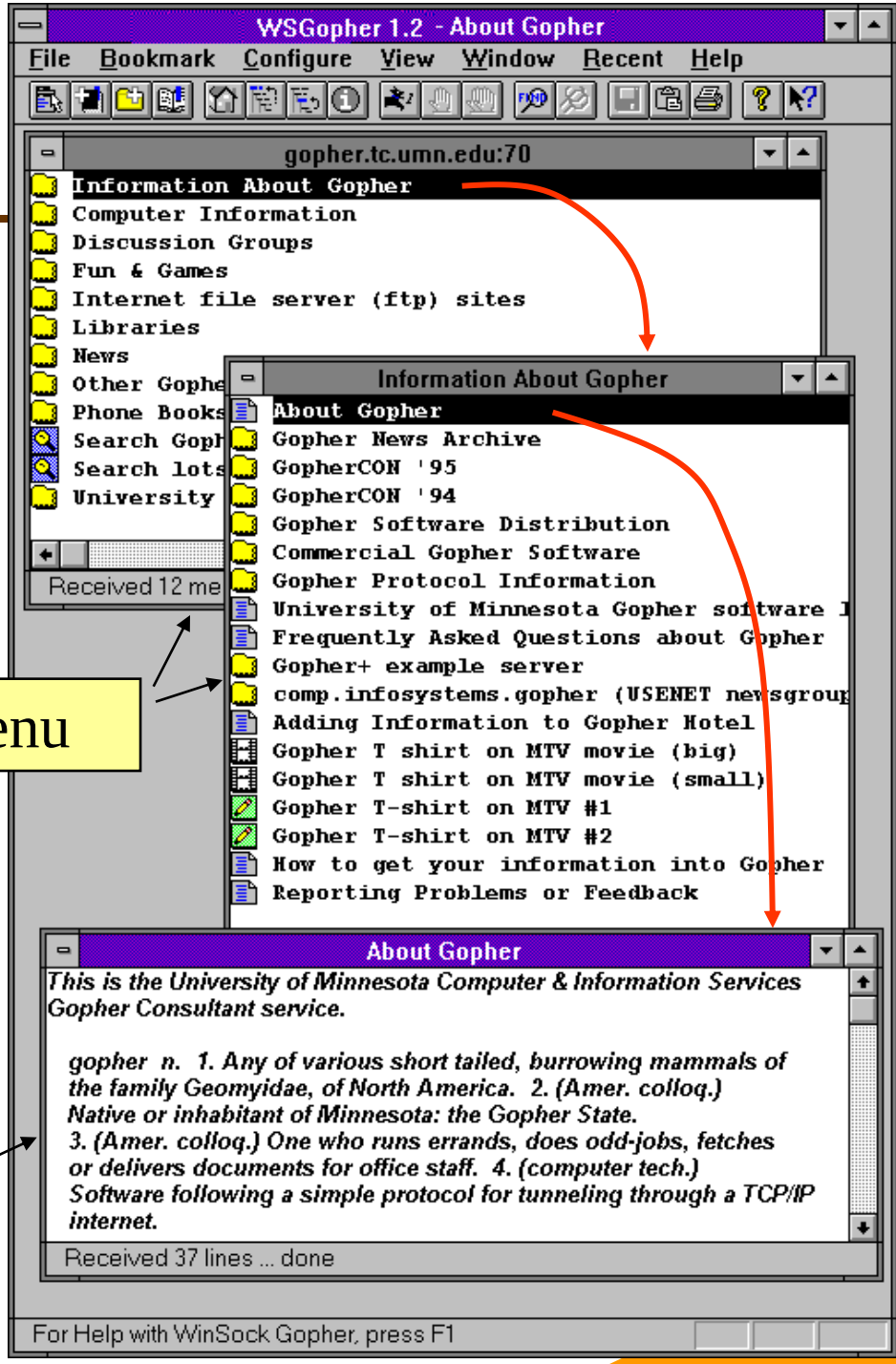


Příklad: menu služby Gopher



Gopher vs. WWW

- v Gopheru se striktně rozlišovalo mezi menu (nabídkou) a obsahem (textem, obrázky, …)
  - obě části jsou specifickými variantami hypertextu
  - odkazy byly pouze v menu, nikoli v textu!!!
- u WWW se mezi menu a obsahem (textem) nerozlišuje
  - zde jakoby splývají !!!



vývoj WWW

- původně:
  - služba jen pro zpřístupnění textů
    - nepodporovala grafiku
      - autoři měli strach ze zneužití, že lidé naskenují tištěné texty a udělají z nich obrázek
  - podpora grafiky se objevila až u NCSA Mosaic
- dnes:
  - podpora celé řady formátů
    - multimediálních
  - původně řešeno pomocí externích aplikací či doplňků
    - helperů, plug-inů
  - později podpora zabudována přímo do browseru jako integrální součást
- původně:
  - skromné prezentační schopnosti
- dnes:
  - velmi dobré prezentační schopnosti
    - schopnost podat informace v "atraktivním obalu"
- původně:
  - WWW byl službou
- dnes:
  - je současně i platformou pro poskytování dalších služeb
    - vyhledávání, adresáře, aplikace .....

proč WWW tak uspěl?

- protože jeho základní princip (hypertext) dobře odpovídá způsobu lidského myšlení
  - protože vychází vstříc lidské slabosti pro „hezký obal“
- protože je neskromný
  - vyžaduje větší přenosové kapacity, výkonnější počítače, ...
- protože dokázal „zlidštit“ Internet a práci v něm, učinit ji mnohem jednodušší a intuitivnější
  - z Internetu „jen pro odborníky“ udělal Internet „pro každého“
- protože dokázal nabídnout atraktivní funkce i komerční sféře
  - stal se předmětem podnikání, místem podnikání, prostředkem podpory podnikání, marketingovým nástrojem

- **protože přišel ve správný okamžik**
  - a porazil svého konkurenta (Gopher)

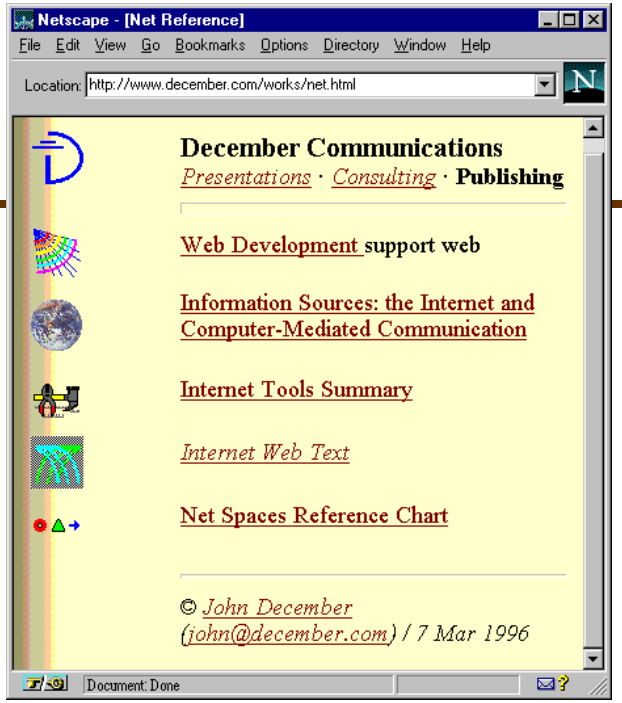
architektura WWW

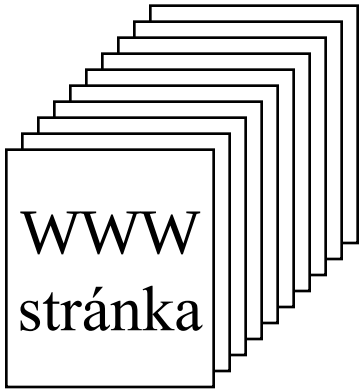
- vychází z architektury klient/server
- předpokládá následující dělbu práce:
  - server (WWW server): uchovává jednotlivé WWW stránky, na (explicitní) žádost je poskytuje svým klientům
  - klient (WWW prohlížeč, browser) si „vyzvedává“ stránky od serverů, zobrazuje je uživateli, zprostředkovává „brouzdání“

pro korektní fungování WWW musí existovat všeobecně dodržované konvence o:

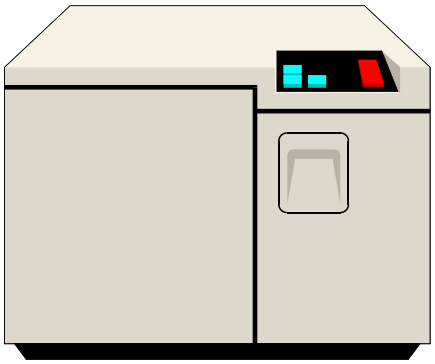
- formátu WWW stránek (zápisu jejich obsahu)
  - toto pokrývá jazyk HTML (HyperText Markup Language)
- způsobu přenosu stránek (mezi serverem a klientem)
  - toto pokrývá protokol HTTP (HyperText Transfer Protocol)

představa



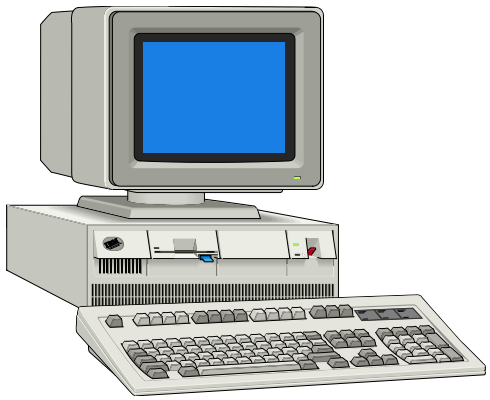


stránky jsou psány v jazyku HTML



WWW server

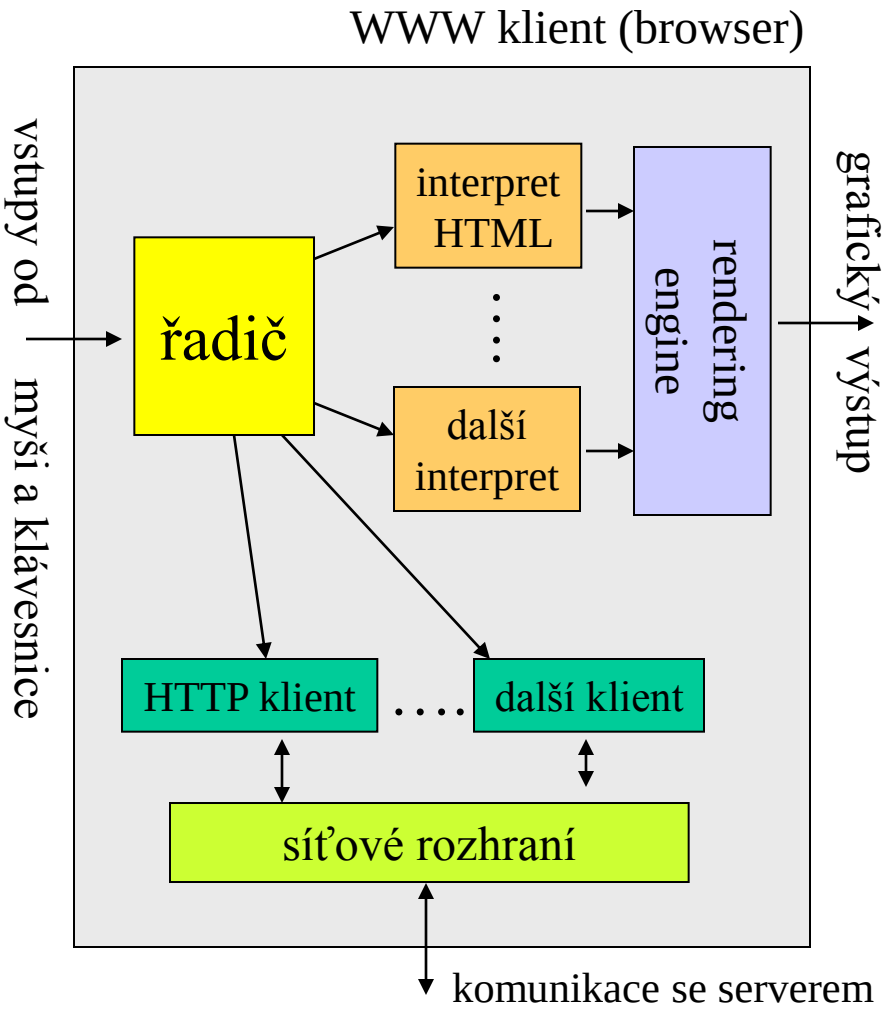
přenos se děje prostřednictvím protokolu HTTP



WWW klient (browser)

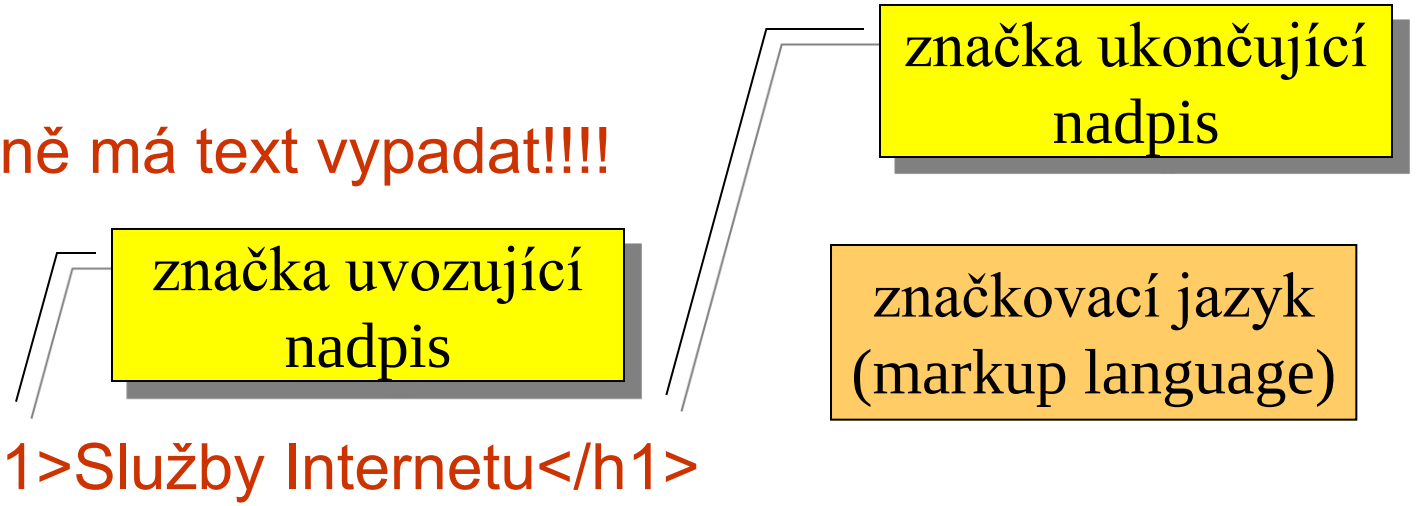
WWW klient - architektura

- klient (browser) rozhoduje o zobrazení (rendering-u) objektů různých typů
  - podle svých grafických možností
- některé formáty dokáže zpracovat sám
  - pro jiné volá externí programy (helpery) nebo "nesamostatné moduly" (plug-in)
- dnešní WWW klienti dokáží pracovat současně i jako klienti dalších služeb
  - hlavně FTP a Gopher
    - obsahují příslušné interprety a protokolové klienty



filosofie jazyka HTML

- říká:
  - čím je text (např. nadpisem, zvýrazněným textem, číslovaným seznamem, ...)
- neříká:
  - jak přesně má text vypadat!!!!
- příklad:



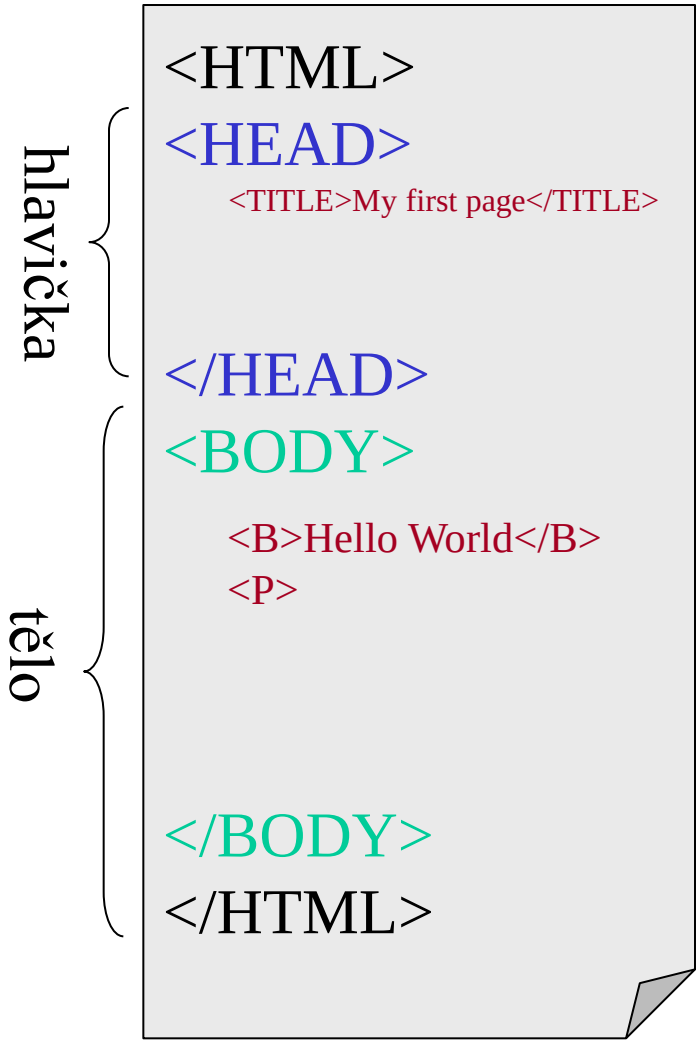
- o tom, jak bude text vypadat (zobrazen), rozhoduje až klient (WWW prohlížeč)!!!!

vývoj jazyka HTML

- původně:
  - pouze jednoduché prezentační možnosti
    - 6 druhů nadpisů, tučné písmo, kurzíva, seznamy ..
  - jednosměrná prezentace
    - žádná možnost jak dopravovat data směrem k serveru
- později:
  - zavedení zpětné vazby
    - formuláře, je možná vyšší interaktivita (zadávání dotazů)
  - větší prezentační možnosti
    - různé druhy písma …..
- cascading style sheets
  - analogie šablon u textových procesorů
    - umožňuje specifikovat vlastnosti objektů samostatně (pomocí šablony) a tu pak aplikovat)
- skripty
  - do HTML kódu je možné vkládat výkonný kód ve zdrojovém tvaru, který je interpretován
    - v rámci HTML lze programovat
- Java aplety a ActiveX objekty
  - výkonné "programy", nikoli ve zdrojovém tvaru

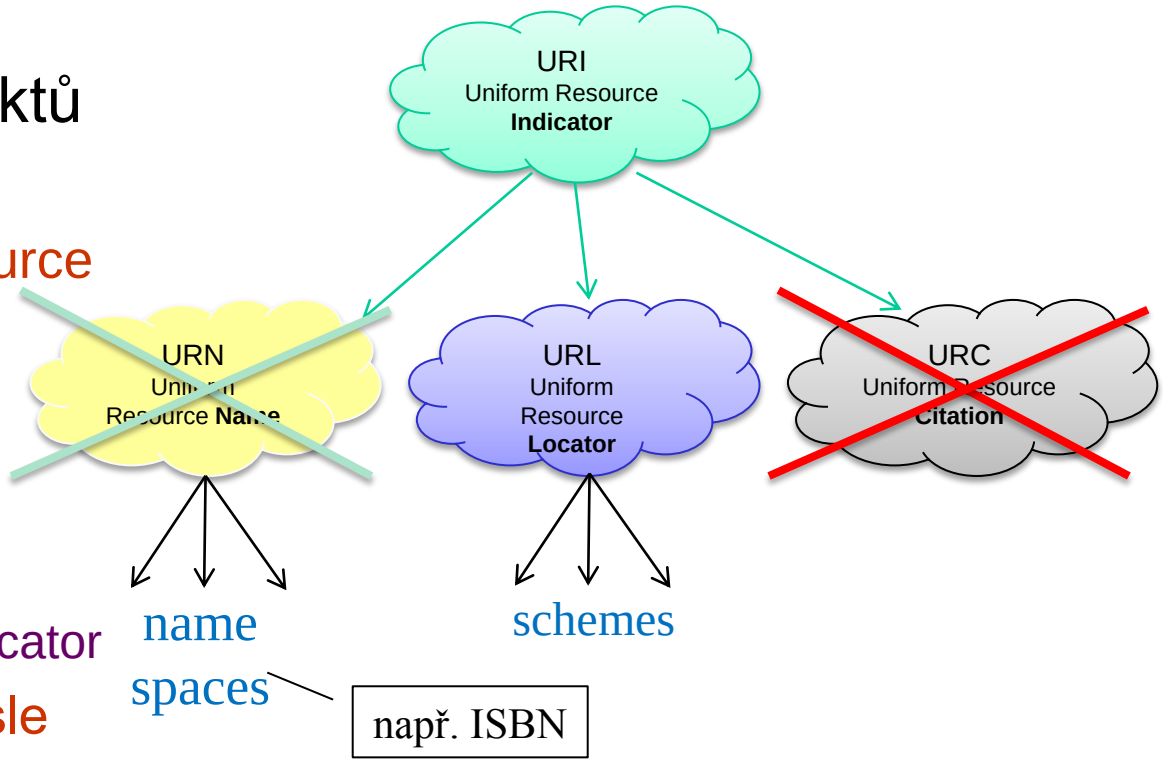
struktura WWW stránky

- hlavička (header)
  - obsahuje "technické" informace
    - titulek (<TITLE>)
    - kódování
    - metadata – např. klíčová slova, informace o autorovi, rating stránek
    - vazba na skripty, CSS, …
    - …..
- tělo (body)
  - obsahuje vlastní HTML kód stránky



URI, URL, URN, URC, …

- snaha o jednotnou identifikaci všech objektů (všech typů)
  - obecně: Uniform Resource Indicator
- lze realizovat podle různých kritérií, např:
  - podle umístění
    - Uniform Resource Locator
  - podle jména (a nezávisle na umístění)
    - Uniform Resouce Name
  - podle citace
    - Uniform Resource Citation



- URC se nikde neujalo
  - URN také jen minimálně
- URI a URL prakticky splynuly
  - a dělí se na schémata

URL (Uniform Resource Locator)

- jsou to jednotné ukazatele, resp. identifikátory všech objektů, na které se lze odkazovat
- obecně:

  <schéma>:<specifická část schématu>
- jeden z tvarů: HTTP schéma

  http://<host>:<port>/<cesta>?<dotaz>

  kde „**http**" říká, jakým způsobem má být k objektu přistupováno
    - tím současně udává jeho typ
  - „**host**" je (standardní) symbolické doménové jméno uzlu, na kterém se objekt nachází
  - "**cesta**" je plné jméno objektu včetně přístupové cesty

jméno	Význam (schéma)
http	hypertext (HTML)
ftp	FTP soubor
file	místní soubor
news	diskusní skupina nebo článek
gopher	Gopher
mailto	zaslání mailu
telnet	remote login

- ne vždy musí být prefix (protokol) uváděn:
  - pokud není uveden, doplňuje se z kontextu
    - WWW browser si doplní http, místo **http://www.earchiv.cz** pak stačí jen **www.earchiv.cz**

příklady URL

- http://www.earchiv.cz/l215/index.php3
  - tato přednáška na WWW
- file:///E:/ARCHIV/l215/index.htm
  - dtto, jako soubor na lokálním disku
- http://jmeno:heslo@www.euronet.nl/~rollo/members/
  - URL na WWW stránku včetně jména hesla pro přihlášení
- telnet://anezka.vc.cvut.cz
  - pro vzdálené přihlášení
- news:cz.net.www
  - diskusní skupina síťových news
- news:8g1ck1$1rp1$1@news.vol.cz
  - jeden konkrétní příspěvek
- gopher://gopher.cesnet.cz/11/.gopherinfo/cz-gophers

ne-ASCII znaky, znaky jako ;/?<>{}[]|&@ apod.

"*divné*" znaky se musí kódovat, např. **%20** (mezera), obecně: "znak %" a číselný kód znaku

dnes již existuje i standard pro kódování "jiných znaků" v celém URL (IRI, Internationalized Resource Identifiers)

protokol HTTP

- je to jednoduchý přenosový protokol
  - přenáší data v textovém tvaru
  - používá transportní služby protokolu TCP
    - není to nutné, lze použít i jiné protokoly
  - server přijímá požadavky na dobře známém portu 80
  - funguje bezestavově
    - dialog s klientem nemění stav serveru
  - navazuje samostatné spojení pro každý objekt v rámci WWW stránky
    - obrázek, ikonu atd.
- komunikace má charakter "žádost-odpověď"
    - klient iniciuje navázání spojení
  - klient pošle svou žádost
  - server pošle odpověď
    - spojení je ukončeno
- žádosti mají formu jednoduchých příkazů
  - označovaných jako **metody**
  - mohou být doplněny dalšími parametry
    - označovanými jako **hlavičky**
- odpovědi mají číselný charakter
  - stejně jako u FTP a SMTP
  - mohou být doplněny posloupností položek (hlaviček) a ev. obsahem WWW stránky

takto to funguje ve verzi HTTP 1.0

verze protokolu HTTP

- HTTP 0.9
  - velmi jednoduchý přenosový protokol
    - minimální možnosti dialogu mezi klientem a serverem
  - neuměl přenášet nic jiného, než jen hypertextové dokumenty
    - server neuměl říci klientovi, jakého typu je poskytnutý výsledek, zda jde o WWW stránku nebo třeba obrázek
      - vždy se bralo jako WWW stránka
    - mohl sloužit pouze pro práci s texty !!!
      - bez obrázků
- HTTP 1.0 (RFC 1945, květen 1996)
  - více možností "vzájemné domluvy" mezi klientem a serverem
    - pomocí hlaviček
  - lze přenášet i jiné typy objektů, než jen WWW stránky
    - pro identifikaci typy přenášeného objektu si HTTP "vypůjčil" MIME typ
  - stále přetrvávají "kapacitní" nedostatky a nevýhody
- HTTP 1.1 (RFC 2068, RFC 2616)
  - řeší nedostatky verze 1.0
  - zavádí:
    - další možnosti dialogu mezi klientem a serverem
      - bohatší hlavičky
    - virtuální WWW servery
      - více WWW serverů na jedné IP adrese
    - persistentní transportní spojení
      - jedno TCP spojení může být využito pro přenos více různých objektů (postupně)
    - pipelining
      - klient může posílat více požadavků za sebou a teprve pak dostávat odpovědi
    - lepší podporu pro cache a proxy paměti
      - zvyšuje to efektivitu
    - možnost výběru verze/varianty obsahu
      - nově je obsažen mechanismus, umožňující vybrat z více dostupných verzí téhož objektu
    - lepší zabezpečení

metody HTTP

- metoda GET
  - požadavek klienta na poskytnutí WWW stránky
  - obecně: GET <URL> HTTP/1.0
    - nebo GET <URL>, pak server nevrací své (HTTP) hlavičky (ale rovnou HTML kód požadované stránky)
- metoda HEAD
  - požadavek na zaslání hlavičky WWW stránky
- metoda POST
  - pošle data na server (a současně také žádá o novou stránku – s odpovědí)
    - používá se při práci s formuláři pro zasílání odpovědí, které mají být dále zpracovány, např. CGI skriptem
      - jinak se pro zpětnou vazbu používá i GET
- PUT, DELETE, LINK, UNLINK
  - nepoužívají se
- …..

hlavičky HTTP

- zprávy protokolu HTTP mohou být doplněny různými hlavičkami
  - upřesňují požadavky či odpovědi
- např.:
  - **Content-Type**
    - specifikuje MIME typ toho, co je v těle zprávy
      - např. Content-Type: text/html; charset=windows-1250
  - **If-Modified-Since <datum>**
    - pouze s metodou GET, stránka je požadována jen je-li novější
  - **Expires <datum>**
    - říká kdy mají být data považována za neplatná (a nemají se dávat do cache). Expires: 0 znamená, že se nemají cacheovat vůbec
  - **Pragma**
    - obecná hlavička, význam závisí na konkrétní implementaci
      - např.: Pragma: no-cache
  - **Authorization**
    - pro zasílání identifikačních údajů (jméno, heslo, …)

- Obecné hlavičky
- Hlavičky doplňující dotaz
- Hlavičky upřesňující odpověď
- Hlavičky, popisující tělo zprávy

hlavičky HTTP

- Referer
  - obsahuje URL, ze kterého pochází požadavek
    - stránka, na které uživatel klikl na odkaz
  - umožňuje zjišťovat, "odkud uživatelé přišli"
- User-Agent:
  - popisuje verzi použitého klienta
    - např. pro statistiku a monitoring
    - lze využít i pro přizpůsobení WWW stránky konkrétnímu typu klienta
- Accept
  - říká, jaké typy odpovědí klient akceptuje
    - může být vyjádřeno i s prioritami (preferencemi)
- Accept-language
  - jaké jazykové verze dokumentu klient akceptuje, včetně preferencí
    - česky, anglicky atd.
- Accept- charset
  - jaké znakové sady klient přijímá
- Accept-encoding
  - např. gzip, compress atd.

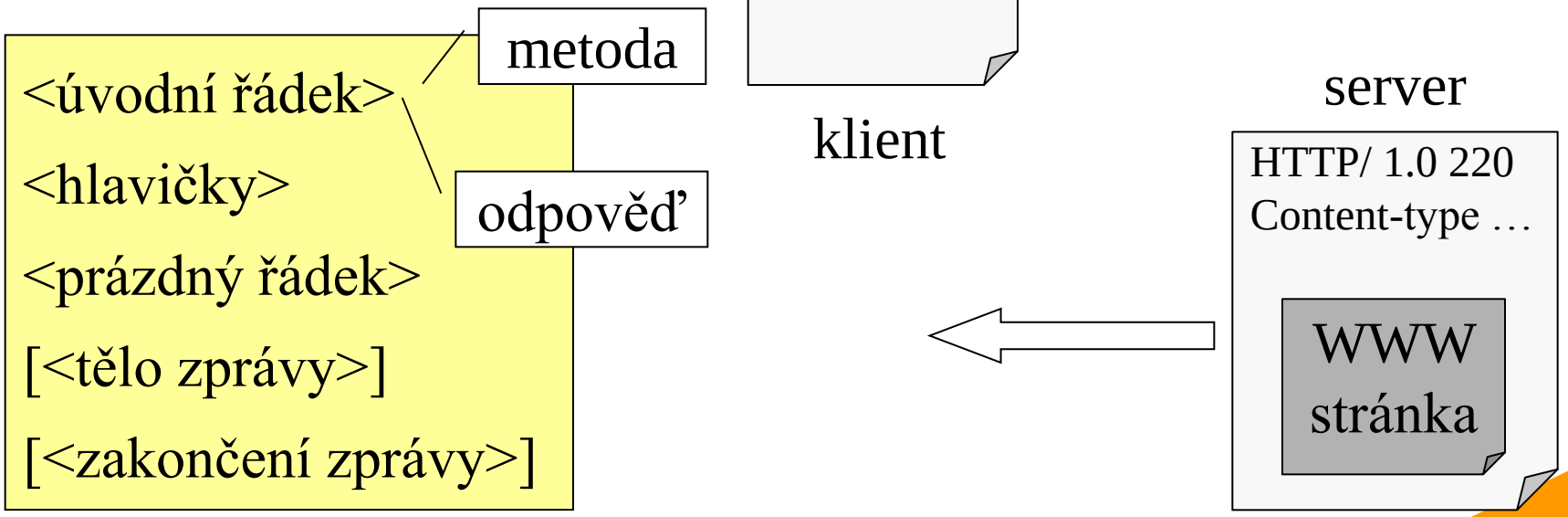
odpovědi HTTP

- odpověď 1xx
  - informační, záleží na aplikaci
- odpověď 2xx
  - kladná odpověď
    - 200 OK, 201 Created, 202 Accepted
- odpověď 3xx
  - očekává se další aktivita ze strany klienta
- odpověď 4xx
  - problém (chyba) na straně klienta
- odpověď 5xx
  - problém (chyba) na straně serveru

- příklady:
  - 400 Bad Request
  - 401 Unauthorized
  - 403 Forbidden
  - 404 Not Found
    - nenalezená stránka – je považováno za chybu uživatele/klienta
  - 500 Internal Server Error
  - 501 Not Implemented
  - 503 Service Unavailable

zprávy HTTP

- požadavky klientů a odpovědi serverů mají povahu zpráv
  - nikoli binárních, ale textových
- struktura a formát zprávy jsou příbuzné zprávám u SMTP pošty
  - jsou členěny na řádky
  - ale znaky jsou 8-bitové
    - na rozdíl od SMTP
- obecný formát zprávy:



příklad dialogu

GET /index.html HTTP/1.0

požadavek klienta

odpověď serveru (2xx)

```
HTTP/1.0 200 OK
Date: Mon, 22 May 2000 21:09:17 GMT
Server: Czech-Net Apache
Content-Length: 546
Last-Modified: Thu, 08 Apr 1999 07:39:05 GMT
ETag: "1386f-222-370c5d19-windows-1250"
Connection: close
Content-Type: text/html; charset=windows-1250
Expires: Thu, 01 Jan 1970 00:00:01 GMT

<html>
    <head>
        <title> …..
```

hlavičky HTTP protokolu (upřesňují odpověď)

poskytnutá WWW stránka

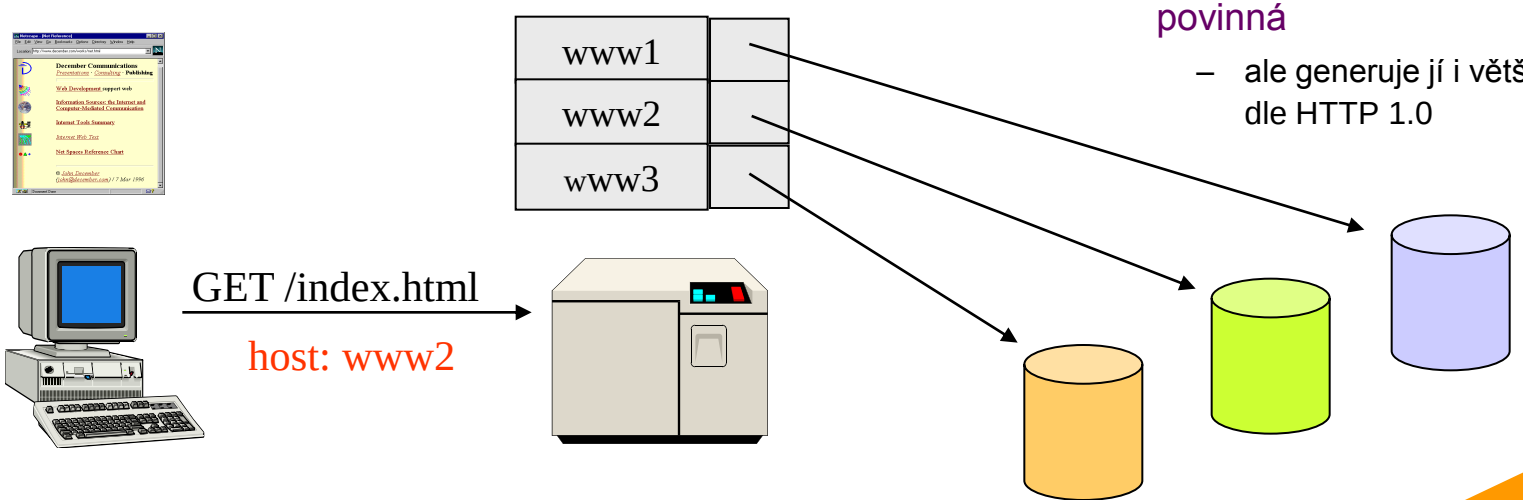
protokol HTTP 1.1 (RFC 2068)

- verze 1.0 otevírá samostatné TCP spojení pro načtení každého jednotlivého objektu
  - např. obrázku
    - obecně pro každý GET
  - je to neefektivní
    - ale je to důsledek snahy o bezestavové fungování
- verze 1.1. umožňuje používat trvalá spojení
  - přetrvávající přes několik požadavků
    - což je podstatně efektivnější
  - podporuje i komprimaci dat přenášených mezi klientem a serverem
  - podporuje i částečné přenosy
    - přenáší se jen určitá část objektu, např. jen část souboru (která se nepřenesla, poškodila atd.)

virtuální WWW servery

- někdy je zapotřebí, aby na jednom počítači "běželo" více samostatných WWW serverů
  - virtuálních WWW serverů
    - každý z nich má vlastní obsah a vlastní URL
- možné řešení: IP-based virtuální WWW server
  - každý virtuální WWW server má svou vlastní IP adresu

- možné řešení: name-based virtuální WWW server
  - všechny virtuální WWW servery (na stejném fyzickém počítači) mají stejnou IP adresu
  - požadavek klienta musí obsahovat hlavičku HOST:
    - např. host: www.earchiv.cz
    - bez této hlavičky není možné rozlišit virtuální WWW server
    - v HTTP 1.1 je tato hlavička povinná
      - ale generuje jí i většina browserů dle HTTP 1.0



cookies (RFC 2109)

- bezestavový charakter komunikace WWW klienta a serveru přináší některé problémy
  - neumožňuje pamatovat si historii předchozí komunikace
    - například uživatelem zvolené kódování, předchozí nákup ve virtuální prodejně atd.
- možné řešení:
  - stavová informace se ukládá do URL
    - server generuje stránky s modifikovanými odkazy URL, aby v nich bylo zakódováno vše potřebné pro "připomenutí"
    - problém je s počátečním krokem
      - uživatel se musí explicitně (a "ručně") přihlásit

lze to vypnout

- jiné řešení: cookies
  - cookie je malý datový údaj, který generuje server, a uchovává jej klient
    - server uloží do cookie vše co si potřebuje pamatovat o aktuální transakci, a pošle cookie klientovi
      - pomocí hlavičky Set-Cookie
    - klient uchová cookie u sebe
    - při příštím požadavku klienta na stejný server je k požadavku přidáno příslušné cookie
      - pomocí hlavičky Cookie
    - server si na základě přijatého cookie "připomene" předchozí historii

Session ID

- představa:
  - server si pamatuje relaci
    - všechna data, která jsou zapotřebí pro udržování relace
    - pamatuje si je (udržuje) ve své databázi
  - server vygeneruje „identifikátor relace“ (Session ID)
    - představa: jde (jakoby) o index do databáze/tabulky všech relací, které si pamatuje
  - server očekává od klienta, že mu relaci připomene
    - server poskytne klientovi Session ID
    - klient při příštím požadavku vrátí („připomene“) toto ID
- možná forma:
  - Session ID se může předávat:
    - v rámci cookie (uživatel přímo nevidí)
    - v rámci URL (URL je „hodně dlouhé“, ID je vidět)
      - http://jizdnirady.idnes.cz/draha/?p=MCUxMjI4NDIlNTQzMzI5NSU1NDM0MDE1JTAlNzA1OSUlMDo0NSUxOjQxJTA4LjA1LjIwMTA

statické, dynamické a aktivní dokumenty HTML

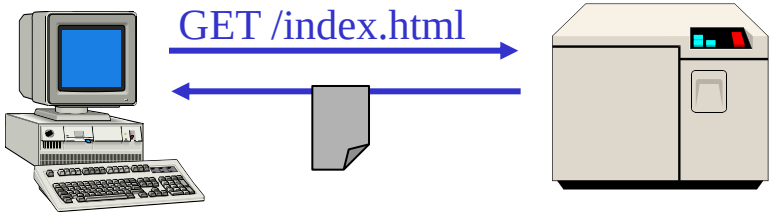
- statický dokument
  - existuje sám o sobě, v neměnné podobě, jako soubor na serveru
    - při poskytnutí klientovi může přesto být transformován, např. překódován do jiné verze češtiny
    - může být indexován fulltextovými vyhledávači
- dynamický dokument
  - neexistuje sám o sobě
    - nemůže být indexován
    - nemá smysl jej cache-ovat
  - je generován až na základě konkrétní žádosti/požadavku
    - vhodnou aplikací
  - nevýhoda: jakmile je dokument jednou vygenerován, už se nemůže měnit
    - může zastarat, např. pokud ukazuje stav rychle se měnícího děje
- aktivní dokument
  - "není dodělán", dotváří se "za chodu"
    - existuje staticky, ale některé jeho části vznikají dynamicky, účinkováním výkonných částí které dotváří výslednou podobu
  - dotváří se u klienta
    - pomocí apletů, prvků ActiveX, skriptů
- výhoda:
  - aktivní dokument se může sám aktualizovat i během svého zobrazení (např. burzovní ticker)

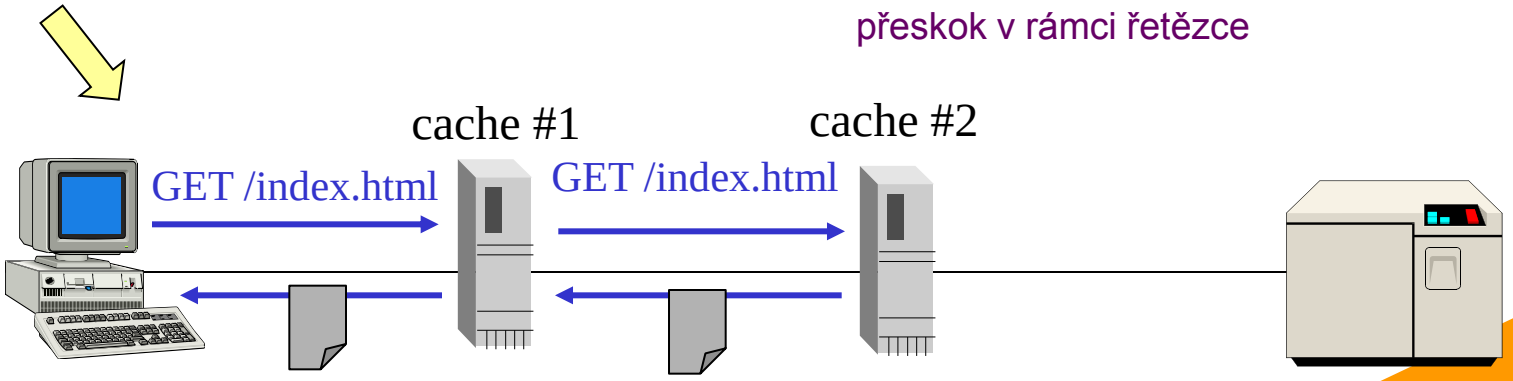
generování dynamických dokumentů

- je třeba zařídit:
  - aby server dokázal spustit externí aplikaci, která vygeneruje dynamický dokument
  - server musí být schopen přijmout výstup této aplikace a vrátit jej klientovi v podobě WWW stránky
  - server musí umět rozlišovat statické a dynamické dokumenty podle URL, a musí vědět jak volat externí aplikace
    - jak jim předávat příslušné parametry
- možnosti realizace:
  - CGI, Common Gateway Interface
    - v zásadě volání externích programů skrze "příkazovou řádku"
      - nezáleží na tom, v čem je program napsán
      - univerzální ale pomalé, pokaždé se znovu spouští nový proces
  - NSAPI (Netscape Server API)
    - umožňuje "připojit" aplikace těsněji k serveru NS
  - ISAPI (Internet Server API)
    - od Microsoftu, výkonné prvky mají formu knihoven .DLL
    - a charakter aplikací nebo filtrů
  - ASP (Active Server Pages)
    - obdoba skriptů a objektů v HTML, ale běží na serveru
  - servlety, Server-Side Include
  - PHP (Personal Homepage Parser)

cache a proxy

- původní představa
  - HTTP zajišťuje přímý dialog 2 entit
    - klienta a serveru
- dnešní realita
  - "v cestě" mezi klientem a serverem může být řada dalších entit
    - cache a proxy serverů
      - kvůli optimalizaci datových toků





- HTTP dialog to komplikuje
  - z dvojice komunikujících entit se stává celý řetězec
    - některé jeho prvky mohou být i transparentní (neviditelné – bez změny nastavení u klienta)
  - HTTP protokol by na to měl pamatovat
    - a umět nějak řídit, či alespoň ovlivňovat chování celého tohoto řetězce
    - HTTP 1.1 to umí lépe než 1.0
  - hlavičky v HTTP:
    - "end-to-end" hlavičky – platí pro celý řetězec
    - "hop-by-hop" hlavičky – platí jen pro daný přeskok v rámci řetězce

možnosti cache-ování

- v rámci klienta
  - klient sám si uchovává objekty ve své cache paměti
    - velmi efektivní: jsou nejrychleji k dispozici
    - málo efektivní: je to cache jen pro něj, nikoli pro jiné klienty/uživatele
- (proxy) caching
  - cache paměť funguje na uzlu uprostřed celého řetězce
    - typicky na proxy serveru
    - může to být společné pro více uživatelů/klientů
- v rámci serveru
  - například pokud generuje stránky dynamicky, jejich uchováváním v cache šetří svůj výkon a zrychluje odpovědi
- úskalí cache-ování
  - objekty nesmí zůstávat v cache paměti příliš dlouho
    - zastarávají, přestávají odpovídat skutečnosti, expirují
  - ale čím déle tam zůstávají …
    - tím se zvyšuje optimalizační efekt celého řetězce
      - častější aktualizace zvyšují režii
- problém
  - jak zvolit správnou (maximální) dobu pro uchovávání objektů v cache pamětech?
- řešení
  - žádné jednoduché a jednoznačné neexistuje

řízení řetězce cache pamětí

- pomocí hlaviček:
  - popisem toho, jaké jsou vlastnosti objektu
    - cache si sama zvolí, jak se vůči objektu zachová
  - explicitními příkazy pro jednotlivé cache
    - přímo se jim řekne/předepíše, jak mají s objektem naložit
      - dodrží to?
- příklady:
  - hlavička Date:
    - říká, kdy byla zpráva vytvořena
  - hlavička Expires:
    - v odpovědi: říká, za jak dlouho objekt zastará (expiruje)

max-age má přednost

- hlavička Cache- Control
  - no-cache:
    - vůbec neuchovávat v cache, vždy si říci o novou verzi
      - totéž jako "Pragma: no-cache"
  - private:
    - objekt je určen jen pro jednoho konkrétního klienta, neměl by být poskytován z cache jiným žadatelům
  - public:
    - opak private
  - max-age <čas>:
    - v žádosti: klient říká, že přijme objekt z cache jen pokud je mladší
    - v odpovědi: server říká, za jak dlouho objekt zastará a měl by být odstraněn z cache
  - only-if-cached:
    - v žádosti: klient chce jen objekt z cache
  - must revalidate:
    - v odpovědi: po expiraci musí být obnoveno