



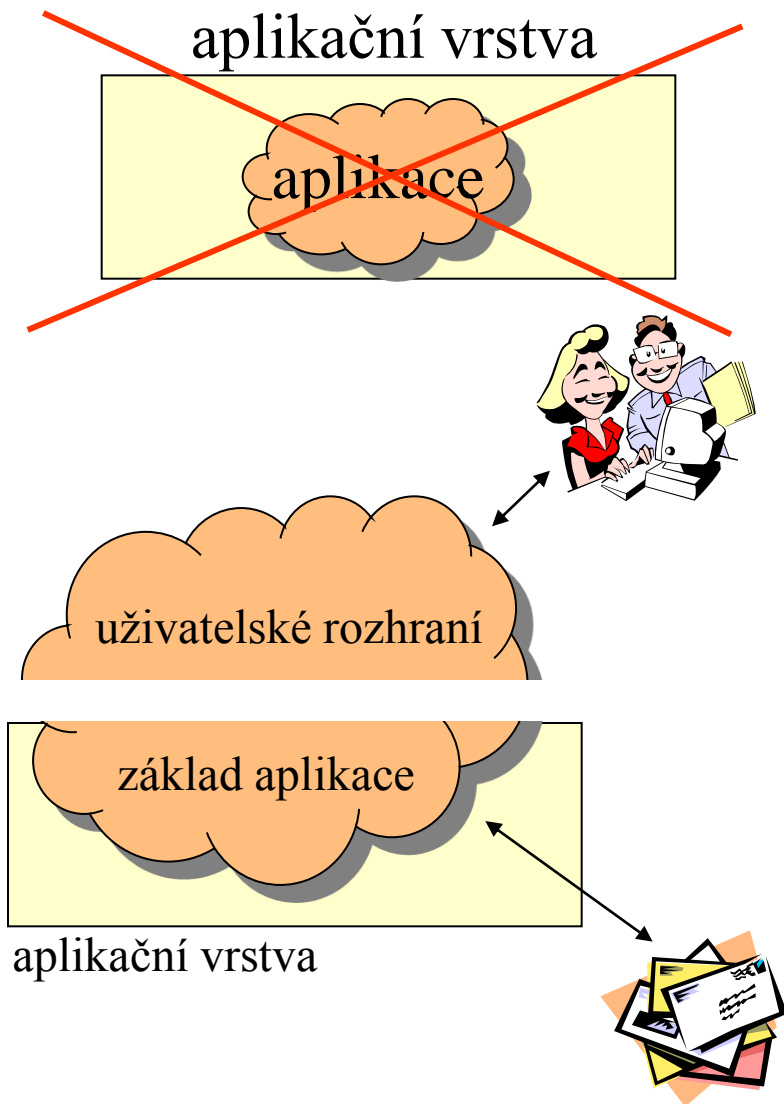
Katedra softwarového inženýrství,
Matematicko-fyzikální fakulta,
Univerzita Karlova, Praha



Lekce 10: Aplikační vrstva

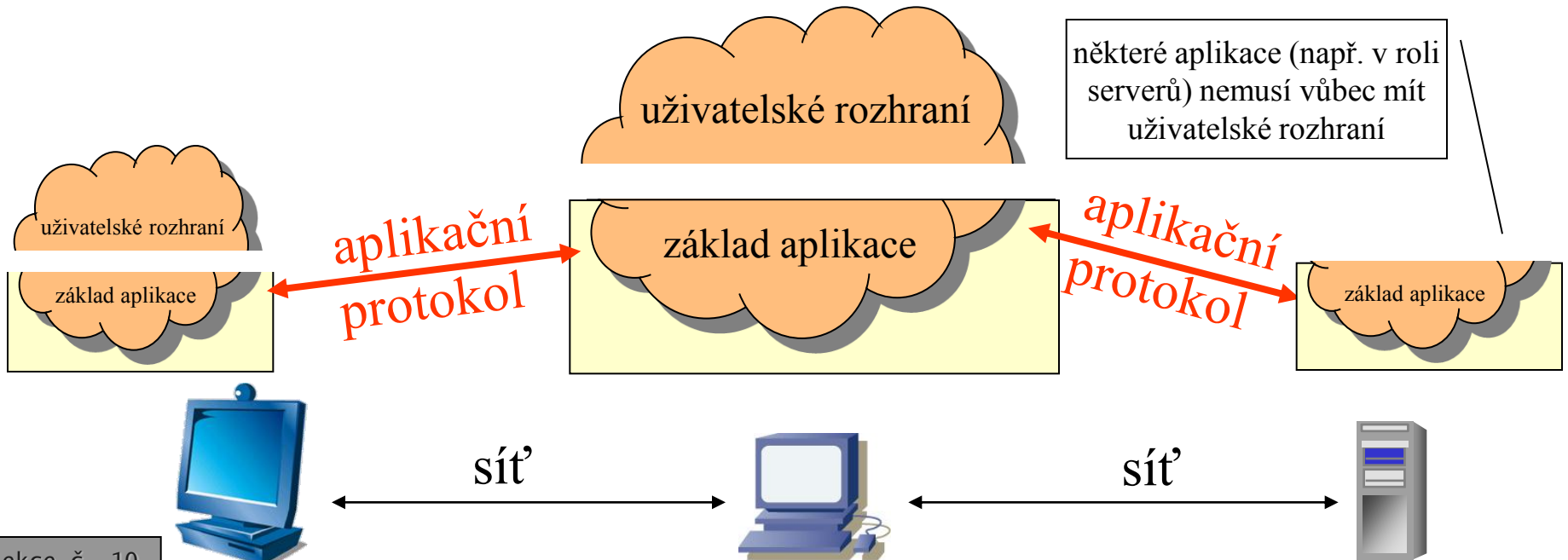
koncepce aplikační vrstvy

- představa, že "v aplikační vrstvě jsou provozovány (celé) aplikace" není správná!!!
 - důvod: bylo by nutné standardizovat celé aplikace
 - včetně uživatelského rozhraní atd.
- místo toho:
 - v aplikační vrstvě je pouze část aplikací
 - "základ aplikace", který spolupracuje s aplikacemi na jiných uzlech
 - tento základ musí být standardizován
 - aby si rozuměl s ostatními "základy"
 - zbytek aplikace je "nad" aplikační vrstvou
 - zejména uživatelské rozhraní
 - někdy se tato část označuje (nesprávně) jako "User Agent"
 - není to dělení "klient/server" !!!!
 - již nemusí být (není vhodné aby bylo) standardizováno
- platí jak pro RM ISO/OSI, tak i pro TCP/IP



koncepce aplikační vrstvy

- "základ" aplikace (v rámci aplikační vrstvy) realizuje nějaká entita
 - nejčastěji: proces
- aplikační entita (proces) komunikuje s jinými entitami v rámci téhož uzlu prostředky "meziprocesové komunikace"
- s aplikačními entitami (procesy) na jiných uzlech komunikuje prostřednictvím **aplikačních protokolů**
 - protokolů aplikační vrstvy
 - jsou šité na míru konkrétním druhům aplikací (např. el. poště, WWW, přenosu souborů atd.)



vývoj aplikační vrstvy

RM ISO/OSI:

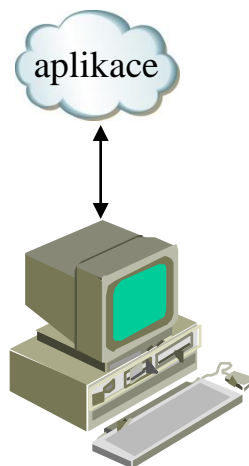
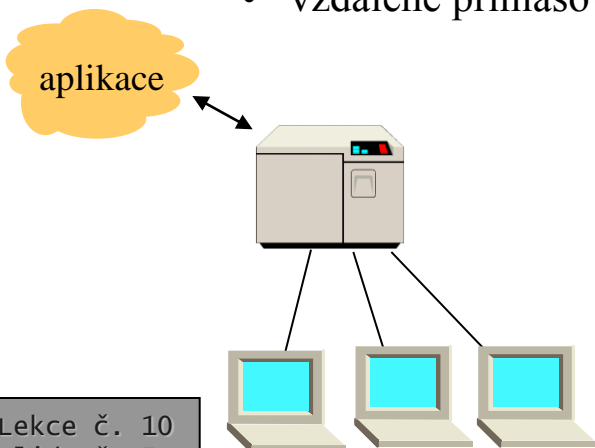
- snaha vytvářet "bohaté" a "dokonalé" aplikační protokoly
 - např.
 - **MOTIS/X.400** – elektronická pošta
 - Message Oriented Text Interchange System
 - **X.500** – adresářové služby
 - **FTAM** – práce se soubory
 - File Transfer, Access and Management
 - **VT** – vzdálené přihlašování
 - Virtual Terminal
 - **CMIP** – správa, management
 - Common Management Information Protocol
 -
 - většina z nich se neujala a nepoužívá se
 - některé ISO/OSI protokoly však přeci jen došly určitého využití
 - např. X.400
 - MS Exchange byl až do verze 2000 primárně založen na X.400
 - např. X.500
 - "odlehčením" vznikl reálně používaný protokol LDAP

rodina protokolů TCP/IP

- postupný vývoj, od jednoduššího ke složitějšímu
 - aplikace vznikaly jako jednoduché, a teprve postupně se obohacovaly
 - rozšiřovalo se také spektrum aplikací
 - "počáteční množina" aplikací:
 - vzdálené přihlašování (**Telnet, rlogin**)
 - přenos souborů (**FTP**)
 - elektronická pošta (**SMTP, RFC 822**)
 - postupně se přidávaly další aplikace
 - sdílení souborů (**NFS**)
 - sdílení informací (**NNTP**)
 - zpřístupnění informací
 - **Gopher**
 - **WWW** (World Wide Web)
 - vyhledávání informací
 - **Archie, WAIS, Veronica**
- dochází ke vzniku "aplikačních platforem"
 - el. pošta a WWW nejsou již jen službami/aplikacemi, ale stávají se platformami, na kterých lze vytvářet nové služby
 - některé původní aplikace časem zanikají
 - např. vyhledávání se stává nadstavbou WWW

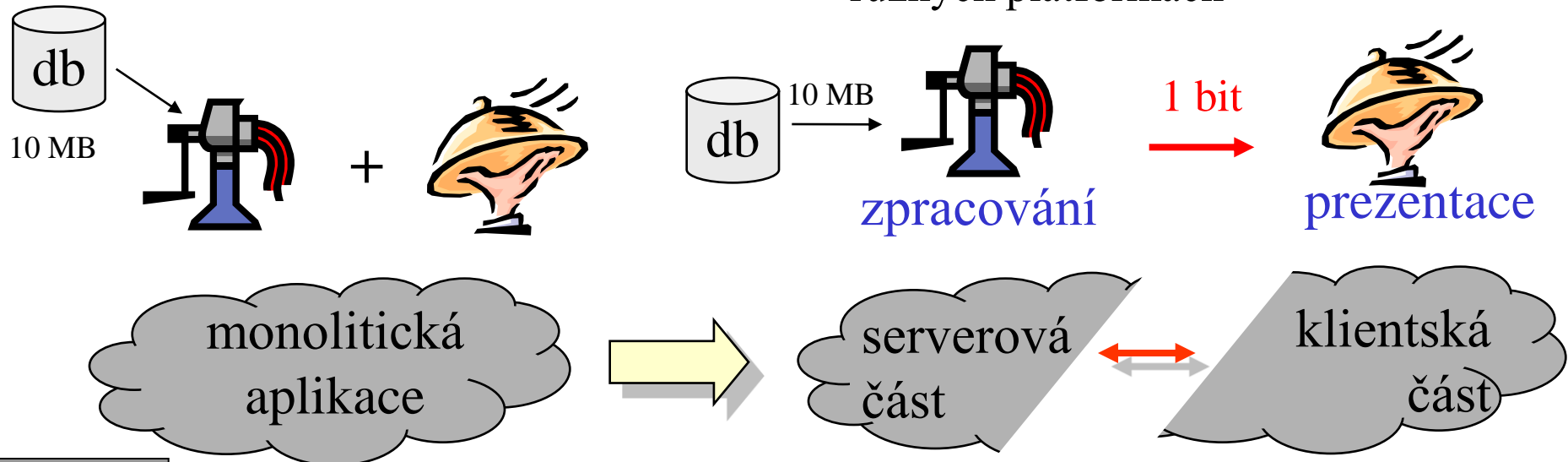
architektura aplikací

- souvisí s tzv. výpočetním modelem
 - ucelenou představou o tom, jak "vypadají" a jak fungují aplikace
 - kolik mají části, kde tyto části běží,
 - kde jsou umístěna data, kde jsou zpracovávána
 -
- výpočetní model se postupně vyvíjí
 - původně: dávkový model
 - dávkové zpracování
 - pak: model host/terminál
 - vzdálené přihlašování
- dnes: monolitické aplikace
 - aplikace si dělá vše sama
 - veškeré zpracování dat
 - vytváří uživatelské rozhraní, komunikuje s uživatelem
 - není rozdělena na více částí
 - (většinou) nespolupracuje s jinými aplikacemi
 - pokud je provozována v prostředí sítě
 - hodí se pro izolované počítače
 - nehodí se (tolik) pro distribuované prostředí – pro síť
 - pokud např. zpracovává větší objemy dat, musí je mít "u sebe"
 - je nutné přenášet velké objemy dat a zpracovávat je jinde, než kde vnikají a jsou standardně uloženy



řešení: model klient/server

- myšlenka:
 - data se budou zpracovávat tam, kde se nachází
 - výstupy pro uživatele se budou generovat tam, kde se nachází uživatel
- musí dojít k rozdělení původně monolitické aplikace na dvě části
 - serverovou část
 - zajišťuje zpracování dat
 - klientskou část
 - zajišťuje uživatelské rozhraní
- klient a server si posílají data představující **dotazy** a **odpovědi**
- pokud se klient a server dobře dohodnou, mohou účinně minimalizovat objem přenášených dat
 - mají výrazně menší přenosové nároky
 - mohou pracovat i v prostředí rozlehlých sítí
- navíc: klient a server mohou stát na různých platformách



představa modelu klient/server



- komunikace mezi klientem a serverem se odehrává stylem: požadavek/odpověď
 - server pasivně čeká, až dostane nějaký požadavek
 - sám se klientům nevnučuje
 - komunikaci iniciuje klient, zasláním požadavku
 - musí být definována **vzájemná komunikace** mezi klientem a serverem
 - komunikační protokol (např. HTTP)
 - musí být definován **formát dat** (zpráva, ...), které si server a klient vyměňují
 - např. jazyk HTML,
- většina aplikací dnes funguje na bázi modelu klient/server
 - **příklad: WWW**
 - WWW server,
 - WWW klient (browser)
 - protokol HTTP,
 - jazyk HTML
 - **příklad: email**
 - poštovní server,
 - poštovní klient
 - protokoly SMTP, POP3, IMAP
 - formát RFC-822, MIME
 -

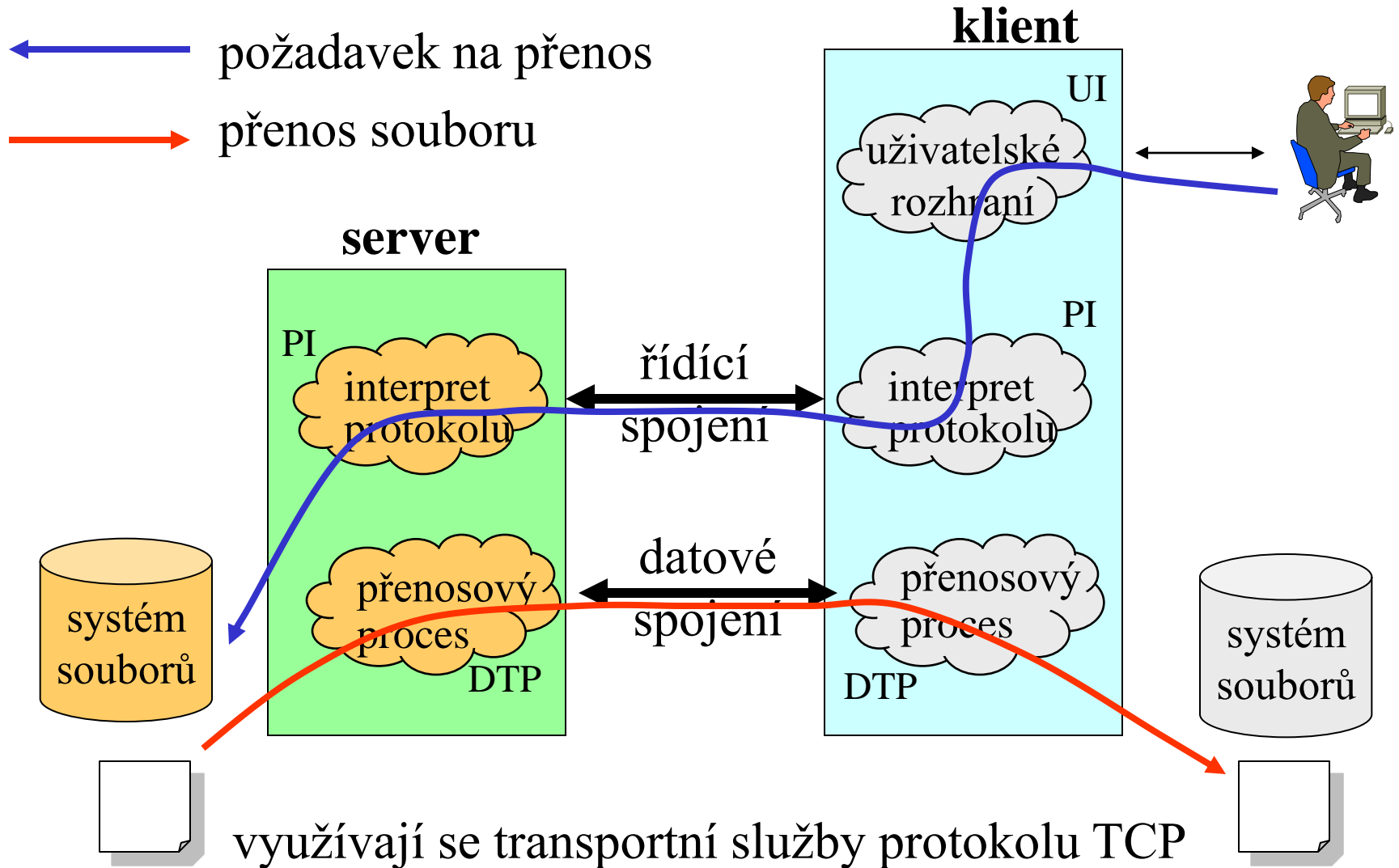
přenos a sdílení souborů

- přenos souborů (*file transfer*)
 - je to služba (realizovaná aplikací)
 - je netransparentní (= rozlišují se místní a vzdálené soubory)
 - je třeba znát umístění vzdálených souborů
 - se vzdálenými soubory se pracuje jinak než s místními
 - pro přesun souborů (z místního umístění na vzdálené) je třeba podnikat explicitní akce
 - příkazy typu "GET", "PUT" atd.
- TCP/IP:
 - nejpoužívanějším protokolem pro přenos souborů je protokol FTP
 - File Transfer Protocol
 - dalším protokolem pro přenos souborů je TFTP
 - Trivial FTP
- RM ISO/OSI:
 - protokol FTAM
 - File Transfer Access and Management
 - realizuje jak přenos souborů, tak i jejich sdílení
- sdílení souborů (*file sharing*)
 - je transparentní (= nerozlišují se vzdálené a místní soubory)
 - není nutné znát umístění vzdálených souborů
 - se vzdálenými i místními soubory se pracuje stejně (jako s místními)
 - pro přesun souborů (z místního umístění na vzdálené a naopak) není třeba podnikat žádné explicitní akce
 - zajišťuje to služba (aplikace) sama
- TCP/IP:
 - nejpoužívanějším protokolem pro sdílení souborů je NFS
 - Network File System
 - dalším je např. AFS
 - Athena File System
 - nově: CIFS
 - Common Internet File System
- RM ISO/OSI:
 - protokol FTAM
- Microsoft, MS Windows:
 - protokol SMB (Server Message Blocks)

FTP – představa a přenos souborů

- FTP implicitně chápe soubor jako dále nestrukturovaný (bez vnitřní struktury) - označováno jako **file structure**
 - proto nepotřebuje "doprovodnou" konvenci o formátu přenášených dat
- implicitně je obsah souboru přenášen jako spojitý proud dat (tzv. **stream mode**)
 - protokol FTP využívá (spolehlivých, spojovaných) transportních služeb protokolu TCP
- implementace vychází z modelu klient/server
 - klient je typicky aplikačním programem
 - server obvykle systémovým procesem (démonem, rezidentním programem apod.)
- návrh protokolu TCP je uzpůsoben možnosti úsporné implementace
 - snaží se nárokovat si systémové zdroje až v okamžiku jejich skutečné potřeby
- zajištění potřebných funkcí v rámci FTP je rozděleno mezi dvě entity:
 - interpret protokolu (**PI, Protocol Interpreter**)
 - přenosový proces (**DTP, Data Transfer Process**)
- interpret protokolu (PI) existuje trvale,
 - přenosový proces (DTP) vzniká až na základě konkrétního požadavku
- používají se dvě různá spojení:
 - **řídící** (pro přenos příkazů)
 - **datové** (pro přenos souborů)

implementace protokolu FTP představa



datové a řídicí spojení

- řídicí spojení iniciuje (navazuje) klient
 - ze svého (dynamicky přiděleného) portu na port 21
 - ruší se až explicitním příkazem
- datové spojení iniciuje (navazuje) server
 - ze svého portu 20 na port klienta, ze kterého bylo navázáno řídicí spojení
 - **passive-mode**: datové spojení nenavazuje server, ale klient
 - kvůli firewallům, které neakceptují žádosti o otevření spojení vedoucí dovnitř na "náhodný" port
- FTP definuje vlastní řídicí jazyk
 - příkazy řídicího jazyka jsou přenášeny řídicím spojením
 - řídicí příkazy mají textovou povahu
- příkazy řídicího jazyka lze rozdělit na:
 - **řízení přístupu** (access control commands) - např. pro zadání uživatelského jména a hesla
 - **nastavení parametrů** přístupu (transfer parameter commands) - např. pro změnu implicitních čísel portů, pro nastavení režimu přenosu apod.
 - **výkonné příkazy** (FTP service commands) - pro vlastní přenos souborů, rušení, přejmenování atd., pro přechody mezi adresáři apod.
- například:
 - **RETR**
 - přenos souboru ze vzdáleného umístění do místního
 - **STORE**
 - přenos z "místního" do "vzdáleného"
 - **LIST**
 - výpis obsahu adresáře
 - **CWD**
 - přechod mezi adresáři

odpovědi na příkazy FTP

- každý příkaz vyvolá alespoň jednu odpověď
- odpovědi mají číselný charakter (s textovým komentářem)
- odpovědi tvoří trojmístné číslo:
 - první číslice vyjadřuje celkový charakter odpovědi
 - druhá číslice upřesňuje odpověď
 - třetí ještě blíže specifikuje
- hierarchický charakter odpovědí vychází vstříc různé inteligenci procesů, které je vyhodnocují
 - “hloupý” klient či server se může spokojit jen s první číslicí
 - “chytrý” klient (server) využije všechny číslice
- stejná konvence (3-místné číselné odpovědi) se používá i u dalších aplikačních protokolů
 - např. u SMTP (elektronická pošta), pro vzájemný dialog serverů
 - u HTTP pro odpovědi serveru na požadavky klienta
 - např. "chyba 404" (stránka nenalezena) – jde o chybu na straně klienta

1xx	předběžná kladná odpověď (akce byla zahájena, budou ještě další odpovědi)
2xx	kladná odpověď (definitivní)
3xx	prozatímní odpověď (jsou nutné další příkazy)
4xx	dočasná záporná odpověď (nepodařilo se, ale je vhodné opakovat)
5xx	trvalá záporná odpověď (nepodařilo se a nemá smysl opakovat)

příklad

navázání (transportního) spojení na uzel charon.isdn.cz, na port 21

- USER earchiv
 - PASS (hidden)
 - CWD /earchiv/
 - RETR users.dat
- Received 411523 bytes in 0.8 secs, (5.15 Mbps), transfer succeeded*
- 220 charon.isdn.cz FTP server ready
 - 331 Password required for earchiv
 - 230 User earchiv logged in
 - 250 CWD command successful
 - 150 Opening BINARY mode data connection for users.dat (411381 bytes)
 - 226 Transfer complete.

definitivní kladná odpověď

prozatím kladná odpověď, nutná ještě další akce

definitivní kladná odpověď

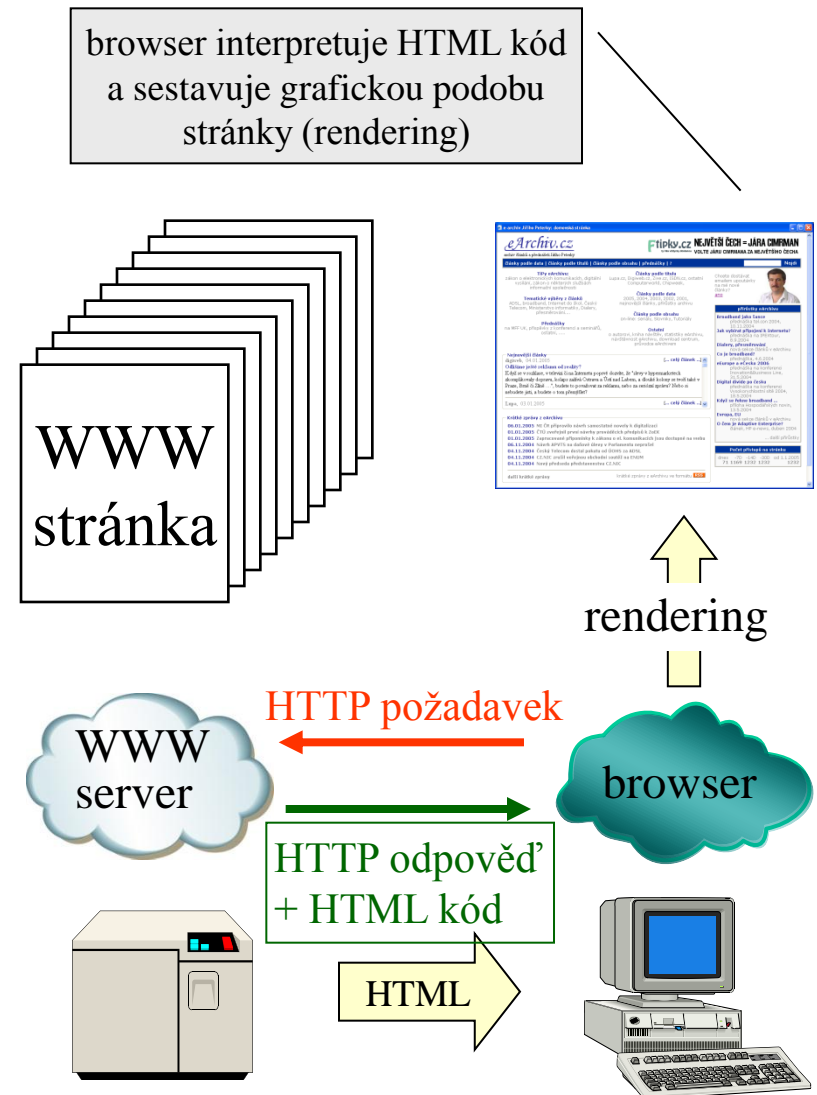
definitivní kladná odpověď

předběžná kladná odpověď, nutná ještě další akce

definitivní kladná odpověď

World Wide Web - architektura

- vychází z modelu 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)



protokol HTTP (HyperText Transfer Protocol)

- 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 bezstavově
 - 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
- odpovědi mají číselný charakter
 - stejně jako u FTP a SMTP
 - součástí odpovědi je i samotný obsah WWW stránky !!!

- každá WWW stránka může obsahovat řadu samostatných objektů
 - 1 x samotný HTML kód stránky
 - n x obrázek
 - další (flashe, audiosoubory, ...)
 - každý objekt může být umístěn na jiném WWW serveru
 - ale nebývá, spíše na stejném
- HTTP verze 1.0:
 - každý objekt na stránce je "získáván" samostatně
 - je pro něj zřizováno samostatné transportní spojení s WWW serverem (na port 80), objekt je vyžádán, přenesen, spojení ukončeno

- HTTP verze 1.1:
 - jsou-li objekty na stejném serveru, jsou "získávány" společně
 - je zřízeno jedno společné transportní spojení s WWW serverem, objekty jsou postupně stahovány, teprve pak je transportní



HTML
kód

eArchiv.cz

Ftipky.cz NEJVĚTŠÍ ČECH JÁRA CIMRMAN
by Vás vítaly doma VOLTE JÁRU CIMRMANA ZA NEJVĚTŠÍHO ČECHA

atd.

metody HTTP

- žádosti WWW klientů (browserů) mají formu jednoduchých příkazů
 - označovaných jako **metody**
 - příklady metod:
 - 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
 - používá se při práci s formuláři pro zasílání odpovědí, které mají být dále zpracovány, např. CGI skriptem
 - » jinak se používá i GET
 - **PUT, DELETE, LINK, UNLINK**
 - nepoužívají se
 - žádosti klientů mohou být doplněny dalšími parametry
 - označovanými jako **hlavičky**
 - příklady hlaviček
 - **If-Modified-Since <datum>**
 - uvádí se např. s metodou GET, a stránka je požadována jen je-li novější
 - **Authorization**
 - pro zasílání identifikačních údajů (jméno, heslo, ...)
 -
- **všechny žádosti klientů začínají "na zelené louce"**
 - server si nepamatuje historii komunikace s daným klientem
 - **důsledek:**
 - komunikace klienta se serverem je bezestavová!!!
 - **výhoda:**
 - požadavky různých klientů mohou být libovolně promíchány, a serveru to nevadí !!!

odpovědi HTTP

- odpovědi WWW serveru mají několik částí:
 - "status odpovědi"
 - používá se stejný systém 3.místných číselných odpovědí jako u FTP a SMT protokolů
 - **1xx**: informační, záleží na aplikaci
 - **2xx**: kladná odpověď
 - např. 200 OK, 201 Created, 202 Accepted
 - **3xx**: očekává se další aktivita od klienta
 - **4xx**: problém (chyba) na straně klienta
 - 400 Bad Request
 - 401 Unauthorized
 - 403 Forbidden
 - 404 Not Found
 -
 - **5xx**: problém (chyba) na straně serveru
 - 500 Internal Server Error
 - 501 Not Implemented
 - 503 Service Unavailable
 -
 - upřesňující hlavičky, například
 - **Content-Type**
 - specifikuje MIME typ toho, co je v "datové části" odpovědi
 - » např. Content-Type: text/html; charset=windows-1250
 - **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
 -
 - "datovou část"
 - např. HTML kód požadované stránky, obrázek, obecně klientem vyžádaný objekt
 - jeho typ je upřesněn v hlavičce Content-type

příklad HTTP dialogu

GET /index.html HTTP/1.0

požadavek klienta

HTTP/1.0 200 OK

odpověď serveru (2xx)

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

Connection: close

Content-Type: text/html; charset=windows-1250

Expires: Thu, 01 Jan 1970 00:00:01 GMT

hlavičky

HTTP protokolu
(upřesňují odpověď)

<html>

<head>

<title>

"datová část"
(poskytnutá WWW stránka)

co je elektronická pošta?

- **je to služba!**
 - může být realizována různými způsoby, v různém prostředí
- existují různé "koncepce" elektronické pošty
 - např. Mail602, ccMail, MS Mail, X.400, SMTP,
 - liší se formátem zpráv, adresami, přenosovými mechanismy, ...
 - obecně jsou vzájemně nekompatibilní
 - pro možnost vzájemné spolupráce vyžadují existenci poštovních bran
- v Internetu se používá tzv. **SMTP-pošta**
 - založená na jedné konkrétní koncepci (na bázi protokolu SMTP a RFC 822)
 - stejná koncepce elektronické pošty může být použita i jinde
 - mimo Internet
 - není proprietární
 - není "vlastněná" žádnou firmou, vychází z plně otevřených standardů



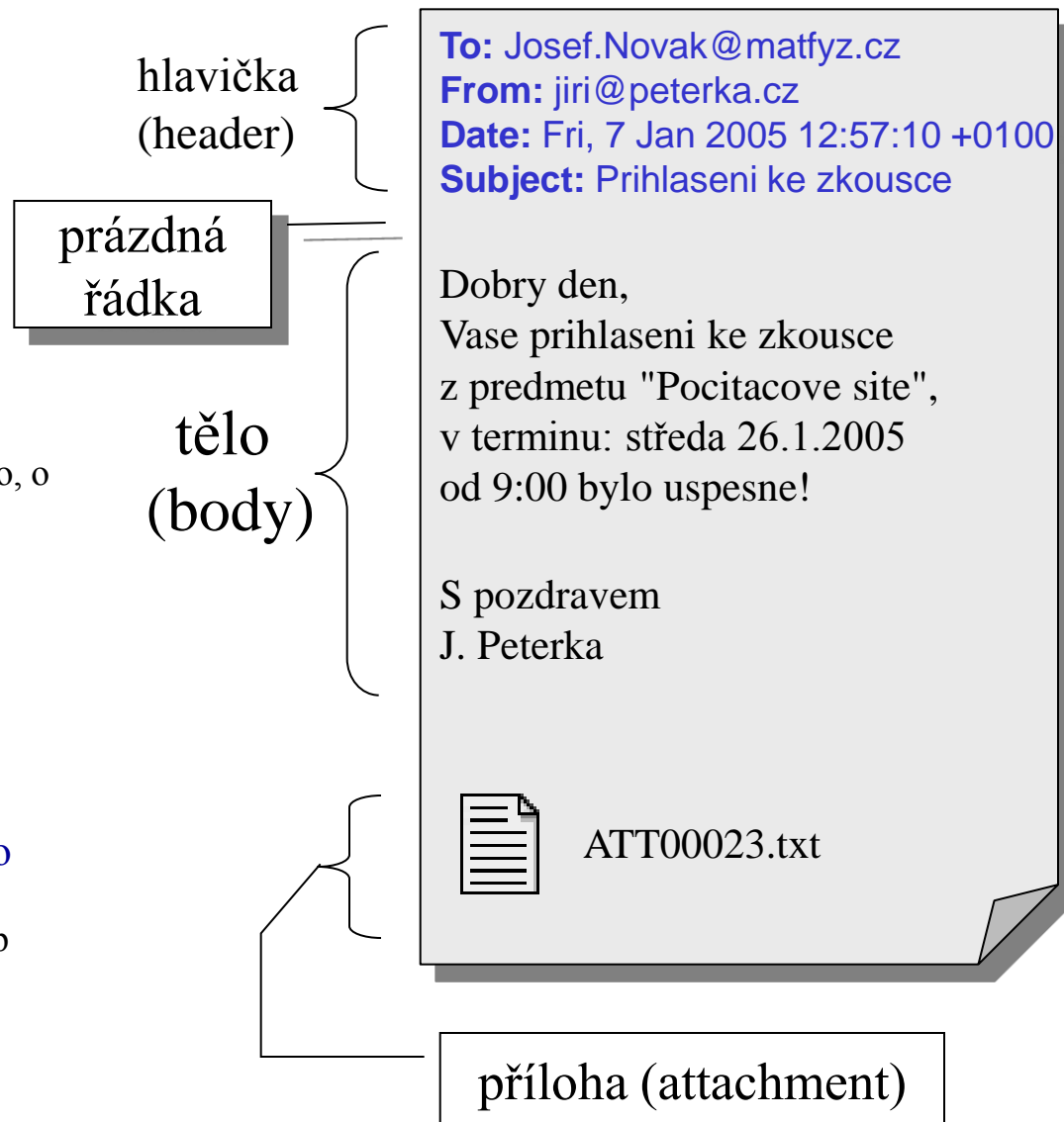
- například SPT Telecom (dnes: Český Telecom) zprovoznil koncem roku 1995 veřejnou elektronickou poštu CZ MAIL, na bázi X.400
 - přenos jednotlivých zpráv byl zpoplatněn
 - první 2 KB zprávy po Evropě stály 8,40 Kč
 - každé další 2 KB stály 4,80 Kč
 - do ostatního světa 15,80 Kč / 8,40 Kč

filosofie a architektura SMTP pošty

- začíná skromně, postupně se obohacuje
 - původně vznikla jako velmi jednoduchá služba
 - jako elektronická obdoba "office memo"
 - původně přenášela jen krátké texty v čistém ASCII tvaru
- další vlastnosti a schopnosti se přidávaly teprve postupně, pokud se ukázala jejich potřeba, např.
 - možnost formátování textu, vkládání obrázků atd.
 - možnost přenosu netextových příloh
 - podpora národních abeced (háčky&čárky)
 -
- a to až po ověření jejich účelnosti a funkčnosti
- vychází z modelu klient/server
 - **poštovní server (mail server)**:
 - v terminologii ISO/OSI: **MTA**, Message Transfer Agent
 - zajišťuje transport zpráv
 - shromažďuje zprávy pro ty účastníky, kteří nejsou momentálně dostupní
 - **poštovní klient**
 - v terminologii ISO/OSI: **UA**, User Agent
 - umožňuje číst, psát a jinak zpracovávat jednotlivé zprávy
 - vytváří uživatelské rozhraní
- standardy el. pošty musí pokrývat
 - **přenos zpráv (mezi servery)**:
 - definuje protokol **SMTP** (Simple Mail Transfer Protocol)
 - **formát zpráv a adres**
 - definuje doporučení **RFC822**
 - **download**
 - stahování zpráv ze schránky na poštovním serveru
 - definuje protokol **POP3, IMAP**
 - **rozšíření (národní abecedy, přílohy, formátování, ...)**
 - definuje standard **MIME**

"anatomie" poštovní zprávy

- Každá zpráva má tyto části:
 - hlavičku (header)
 - tělo (body)
 - volitelně: přílohu (attachment)
- Hlavička obsahuje:
 - adresu příjemce (příjemců)
 - adresu odesílatele
 - datum vzniku/odeslání
 - předmět zprávy (subject)
 - jednořádkový, výstižný popis toho, o co jde
 - další atributy zprávy
 - např. naléhavost, požadavek na potvrzení příjmu,
- Tělo
 - obsahuje vlastní text zprávy
- Příloha:
 - v zásadě cokoli, co lze "zabalit" do podoby souboru
 - např. datový soubor, zvukový klip apod.



RFC822 vs. SMTP

- **Představa:**
 - *“zpráva je list papíru, který se vloží do obálky a teprve ta se přenáší”*
- RFC 822 definuje, co a jak má být napsáno na **“listu papíru”**
- SMTP definuje **“obálku”** a způsob jejího přenosu (i co má být napsáno na této obálce)
 - některé z položek hlavičky “listu” jsou kopírovány na “obálku”, mj. adresa příjemce a odesilatele
- SMTP je přenosovým mechanismem pro přenos zpráv (“obálek”)
 - využívá spolehlivých přenosových služeb protokolu TCP (ale může být implementován i nad jinými spolehlivými přenosovými protokoly)
 - chápe přenášená data jako text
 - členěný na řádky pomocí CR+LF
 - **tvořený 7-bitovými ASCII znaky**

To: Josef.Novak@matfyz.cz
From: jiri@peterka.cz
Date: Fri, 7 Jan 2005 12:57:10 +0100
Subject: Prihlaseni ke zkousce

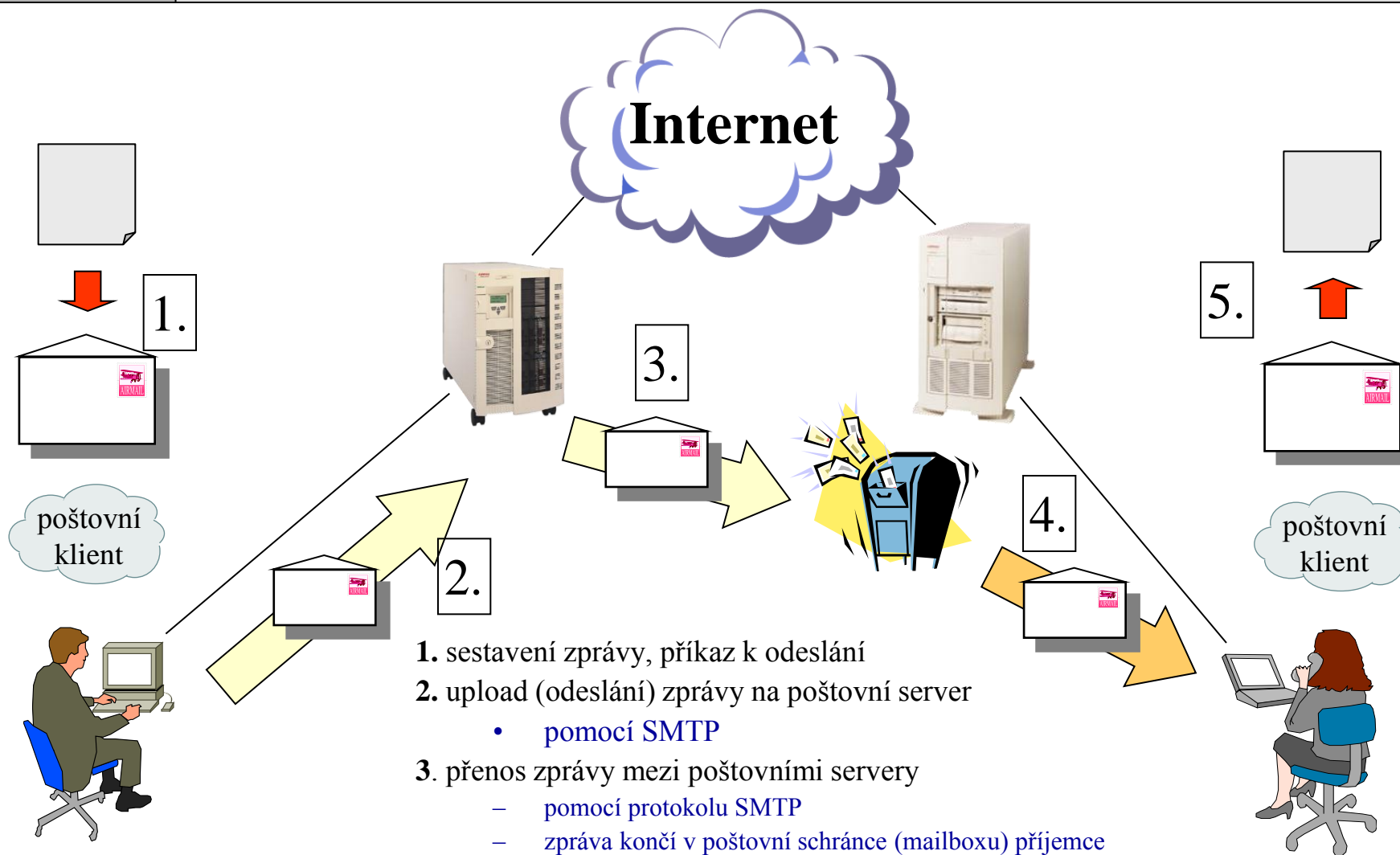
Dobry den,
Vase prihlaseni ke zkousce
z predmetu "Pocitacove site",
v terminu: streda 26.1.2005
od 9:00 bylo uspesne!

S pozdravením



from: jiri@peterka.cz
to: josef.novak@matfyz.cz

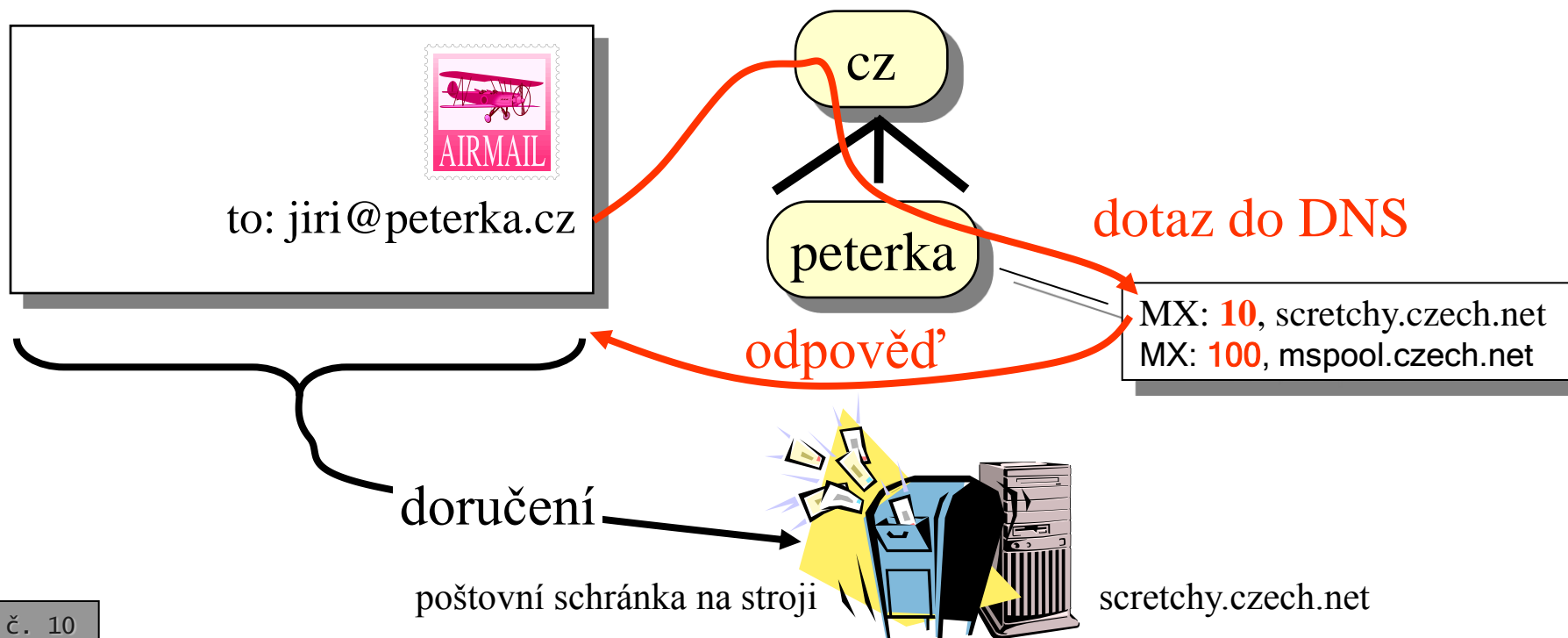
představa přenosu zpráv



1. sestavení zprávy, příkaz k odeslání
2. upload (odeslání) zprávy na poštovní server
 - pomocí SMTP
3. přenos zprávy mezi poštovními servery
 - pomocí protokolu SMTP
 - zpráva končí v poštovní schránce (mailboxu) příjemce
4. stažení zprávy z poštovní schránky do poštovního klienta
 - pomocí protokolu POP3
5. čtení přijaté zprávy, v rámci poštovního klienta příjemce

doručování podle MX záznamů

- emailové adresy dnes mají nejčastěji tvar alias@doména
 - např. jiri@peterka.cz
- jak příjemce pozná, na který SMTP server má zprávu doručit
 - k dispozici má pouze jméno domény
- řešení:
 - pro každou doménu je definován tzv. MX (mail exchanger) záznam
 - definuje jeden (nebo více) SMTP serverů, které přijímají poštu pro danou doménu



doručování zpráv – SMTP pošta

- odesílatel (poštovní klient odesílatele) sám typicky nedoručuje zprávy
 - zná "nejbližší" poštovní server, a tomu předá zprávu k odeslání/doručení
 - "nejbližší" = ten, který má poštovní klient uvedený ve vlastní konfiguraci
 - zpráva se předává pomocí protokolu SMTP
- teprve "nejbližší" poštovní server se stará o doručení převzaté zprávy
 - hledá SMTP server, kterému by měl zprávu předat
 - nejdříve hledá podle MX záznamů v DNS
 - pokud se nedaří určit přijímající server z DNS, snaží se odesílatel interpretovat část adresy za zavináčem jako jméno konkrétního počítače
 - odesílající SMTP server naváže spojení s přijímajícím serverem
 - transportní spojení směřuje na port 25 (kde čeká SMTP server)
 - spojení využívá transportní protokol TCP
 - následuje "SMTP dialog"
 - obě strany si předávají důležité "identifikační" údaje
 - mj. údaje, představující "nápis na obálce"
 - teprve pak je přenesena vlastní zpráva ("list")
 - příkazy protokolu SMTP mají textový charakter
 - např. HELO, EHLO, RCPT, ...
 - odpovědi v SMTP jsou zásadně číselné
 - trojmístné – používá se stejná konvence jako u FTP a HTTP
 - 1xx – předběžná odpověď
 - 2xx – definitivní odpověď
 - 3xx – prozatímní odpověď, nutné další akce
 - 4xx – dočasná chyba, lze zkoušet znovu
 - 5xx – trvalá chyba, nemá smysl zkoušet znovu
- vlastní dialog má i protokol POP3 pro stahování pošty z poštovních schránek
 - POP3 server poskytuje své služby na portu č. 110

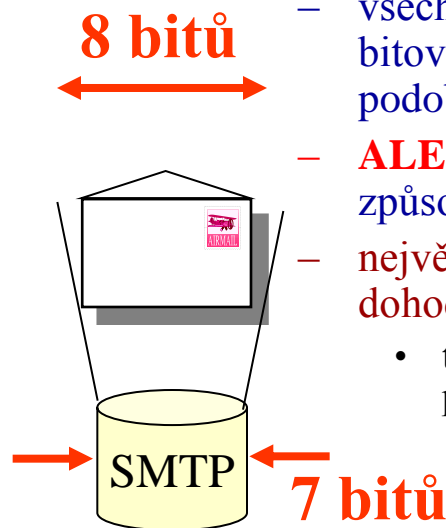
SMTP dialog - průběh

- navázání transportního spojení na port 25 (uzlu scretchy.czech.net)
- 220 scretchy.czech.net SMTP service ready
- HELO smtp.post.cz
- 250 scretchy.czech.net hello smtp.post.cz
- MAIL FROM: <nekdo@post.cz>
- 250 sender ok
- RCPT TO: <jiri@peterka.cz>
- 250 recipient ok
- RCPT TO: <jirka@peterka.cz>
- 250 recipient ok
- DATA
- 354 Enter mail, end with "." on a line by itself
- { hlavička zprávy dle RFC 822 }
- { tělo zprávy dle RFC822 }
- . { tečka jako zakončující znak }
- 250 mail accepted { ukončení přenosu dat }
- QUIT
- 221 scretchy.czech.net { ukončení transportního spojení }



netextové přenosy v SMTP poště

- Původně:
 - SMTP pošta byla určena jen pro přenos krátkých textových zpráv v "čistém ASCII"
 - bez háčků&čárek, bez formátování, různých druhů písma
 - **přenosové mechanismy (protokol SMTP) jsou koncipovány tak, aby garantovaly přenos textových zpráv složených ze 7-bitových znaků**
 - **není stanoveno co se má stát, když znaky budou 8-bitové !!!**
- Problém:
 - pokud se někdo pokusí přenést něco jiného než 7-bitové znaky, není garantováno jak to dopadne
 - může to dopadnout dobře
 - ale: "nejvyšší bity" mohou být ořezány (nastaveny na 0) apod.
- problém je s přílohami
 - pokud by k textové zprávě byl přiložen datový soubor, nemusel by "projít"
 - datový soubor je obecně tvořený 8-bitovými byty
- problém je i s národními abecedami
 - nelze používat znaky národních abeced, protože ty je nutné kódovat do 8 bitů
- problém je i s formátováním
 - formátovací znaky jsou také 8-bitové
- princip řešení:
 - všechno co je 8-bitové se převede na 7-bitové, přeneše a pak zase vrátí do původní podoby
 - **ALE:** toto lze učinit mnoha různými způsoby
 - **největší problém je v tom, aby se lidé dohodli na společném postupu**
 - tak aby příjemce vždy věděl, co a jak má provést s obdrženou zprávou

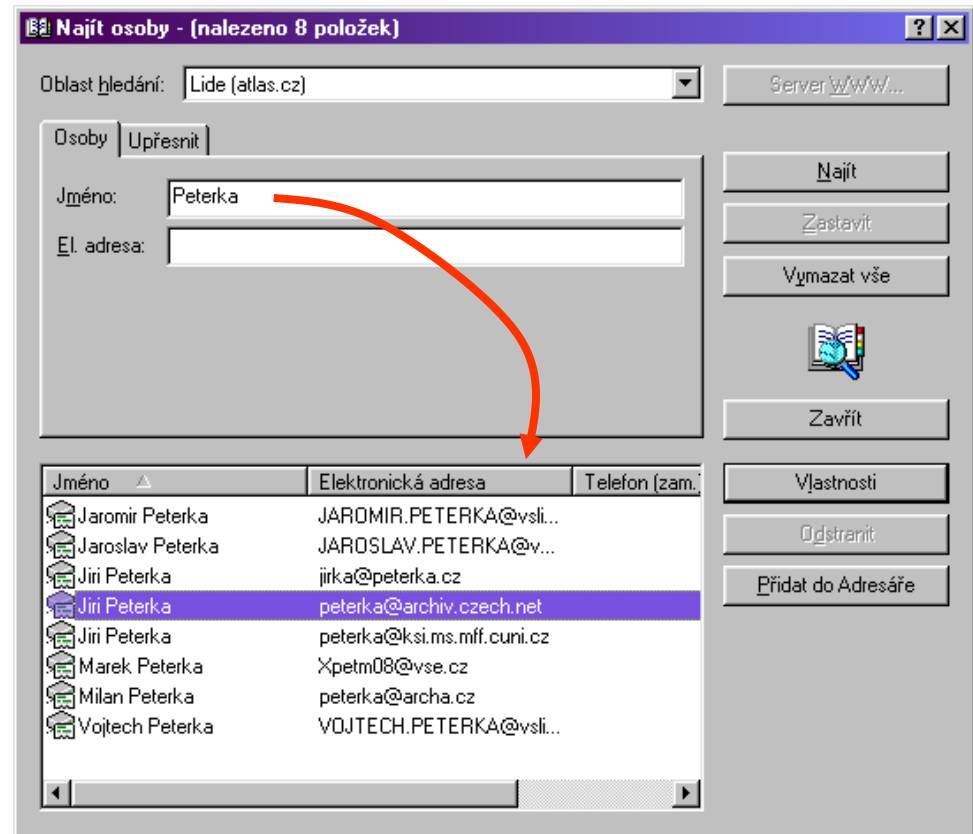


řešení "netextových" přenosů v rámci SMTP pošty

- "nesystematická" řešení:
 - týkají se pouze "přibalování" příloh
 - UUENCODE
 - varianta "přibalování" příloh, pocházející ze světa Unixu
 - BinHex
 - varianta pocházející ze světa počítačů Macintosh
 -
- systematické řešení: standard **MIME**
 - **M**ultipurpose **I**nternet **M**ultimedia **E**xtensions
 - je podporován většinou novějších poštovních klientů
 - umožňuje bezproblémovou práci s přílohami
 - jedna zpráva může mít i více příloh,
 - přílohou může být cokoli, co lze "zabalit" do podoby souboru
 - umožňuje psát česky
 - v těle zprávy, předmětu zprávy i v komentářových částech adres!!!
 - umožňuje provázání poštovního klienta s aplikacemi
 - tak aby uživateli stačilo kliknout na ikonku s přílohou, a klient věděl co má s přílohou udělat (jak ji "vybalit" a kterému programu ji předat)
- co všechno definuje standard MIME?
 - kódování
 - 2 způsoby převedení 8-bitových dat do 7-bitové podoby:
 - Quoted Printable a Base64
 - typování dat
 - zavádí tzv. MIME type (je dvousložkový), aby bylo možné definovat co jsou data zač a bylo možné odvodit, jak mají být zpracována
 - např. text/HTML, image/gif
 - rozšíření formátu zábravy
 - zavádí rozšíření formátu dle RFC822, tak aby mohly být ve zprávě vyjádřeny informace související s přílohami, kódováním atd.
 - zavádí nové položky do hlavičky
 - umožňuje aby tělo zprávy mělo více složek
- standard MIME je typickým příkladem vývoje aplikací v rámci TCP/IP
 - nejprve jsou vyvinuty jednoduché aplikace
 - když se aplikace uchytlí a uživatelé si vznikne potřeba jejich zdokonalení, toto se připraví

další aplikační protokoly TCP/IP

- IMAP
 - Internet Message Access Protocol
 - umožňuje pracovat se zprávami přímo ve vzdálené poštovní schránce
 - není nutné je stahovat, jako u POP3
- S/MIME (secure MIME)
 - rozšíření MIME o bezpečnostní prvky
- NNTP
 - Network News Transfer Protocol
 - "síťové noviny", služba USENET
- Telnet
 - vzdálené přihlašování
- LDAP
 - Lightweight Directory Access Protocol
- NTP
 - Network Time Protocol
-



(příklad využití protokolu LDAP pro vyhledání adresy ve veřejném adresáři)

historické služby/aplikace TCP/IP: Gopher



gopher =

- zool.:pytlonoš kanadský
- americký sysel
- Minnesoťan (přezdívka)

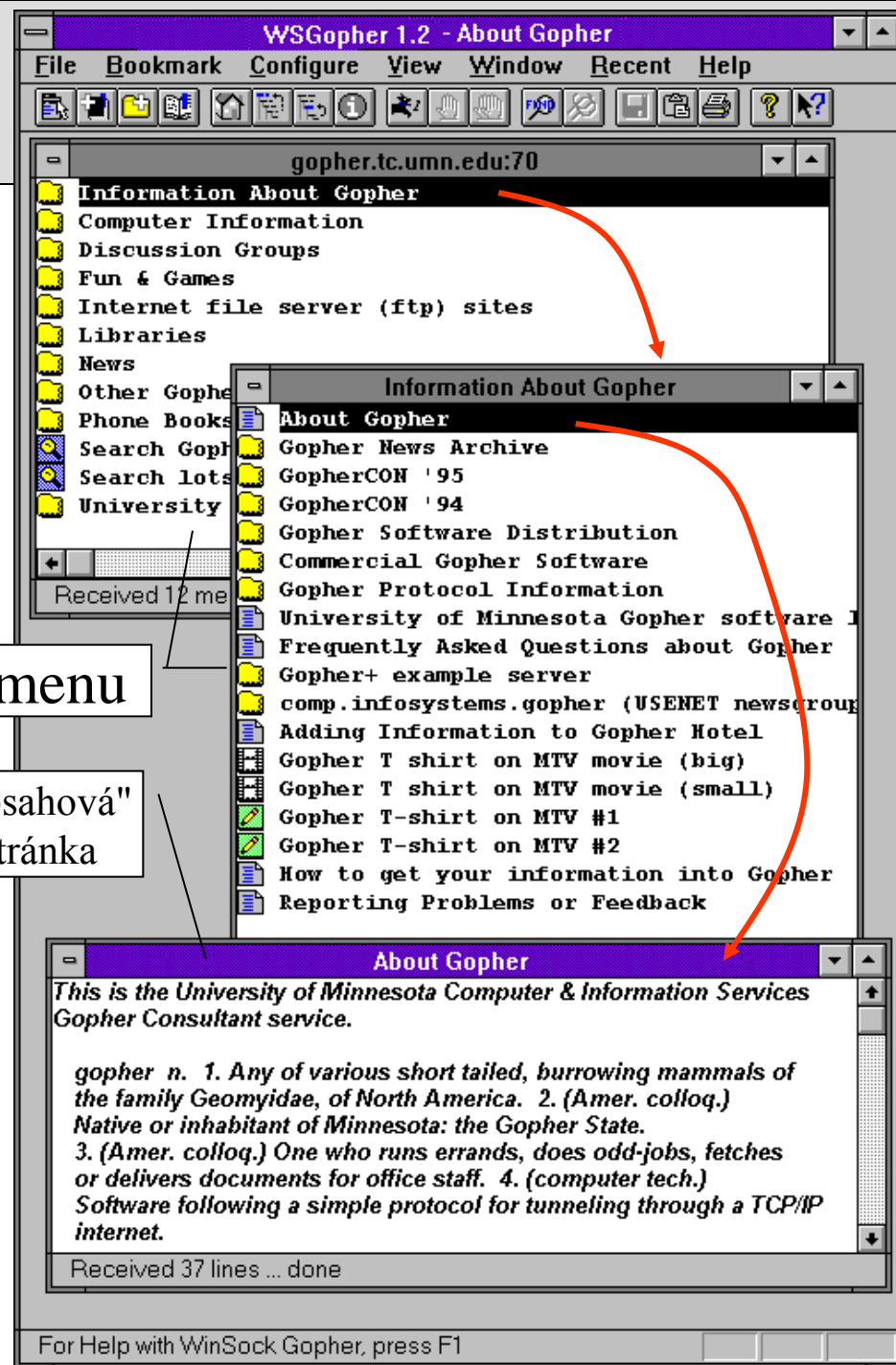
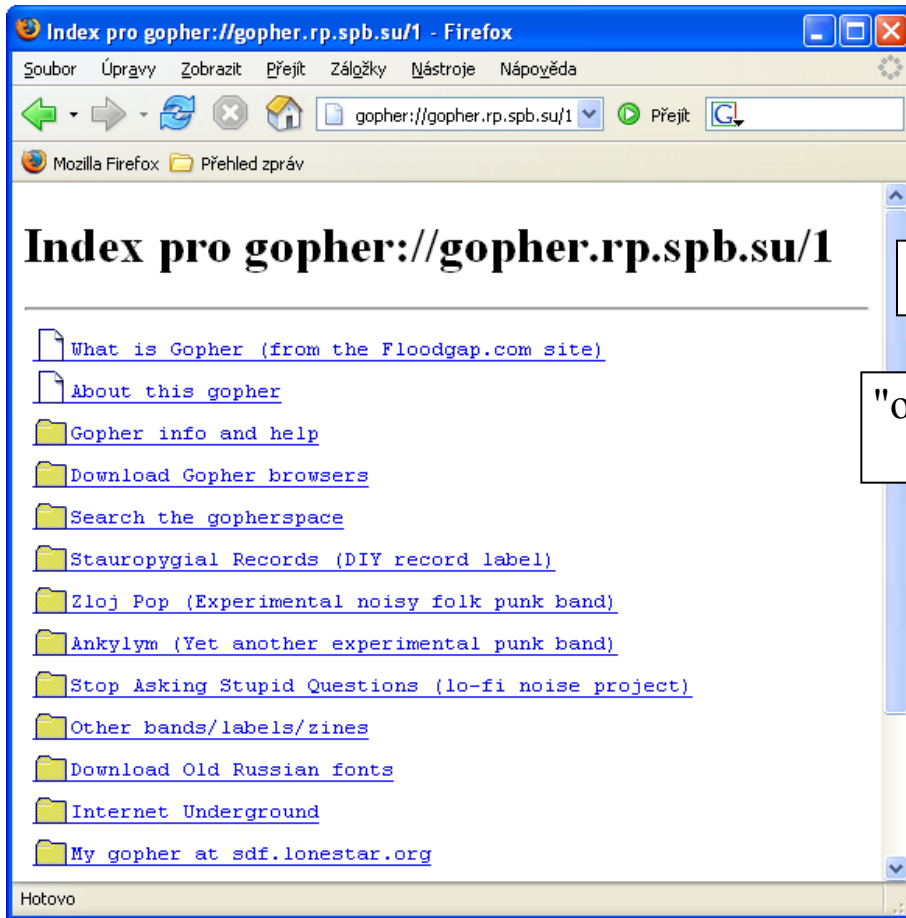
nebo je to odvozeno od "to GO FOR information"?

- Gopher prohrál v souboji s WWW
 - nebyl tolik "sexy"

- Gopher byl vyvinut na University of Minnesota, USA
- je to služba pro zpřístupnění informací
- uživatelé poskytují nabídku ve formě menu
 - jednotlivé položky menu jsou uspořádány lineárně
 - položky jsou textové (i celé menu)
 - položka může představovat:
 - soubor (text, obrázek,)
 - odkaz na jiné menu
 - přechod (bránu) do jiné služby či aplikace

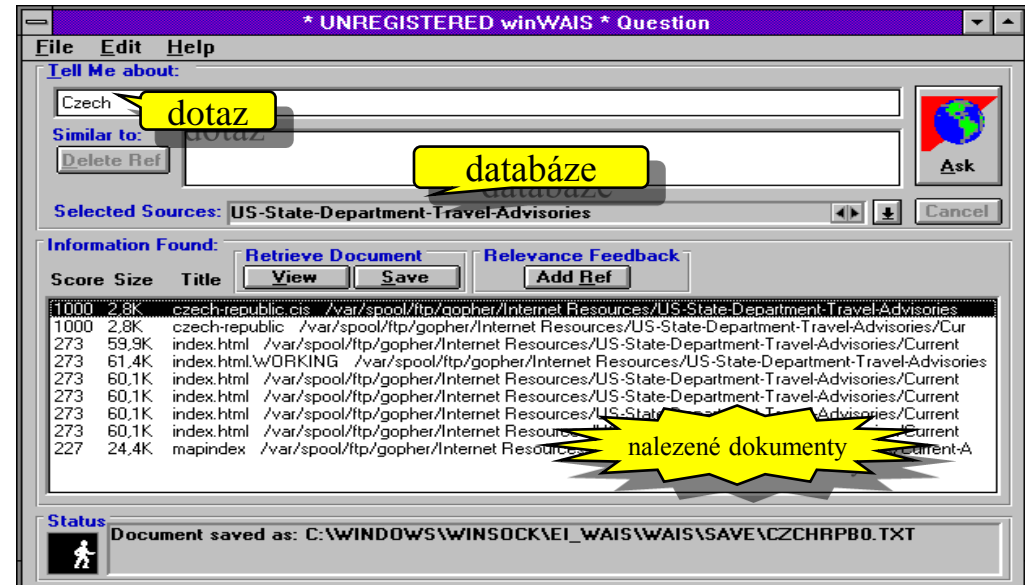
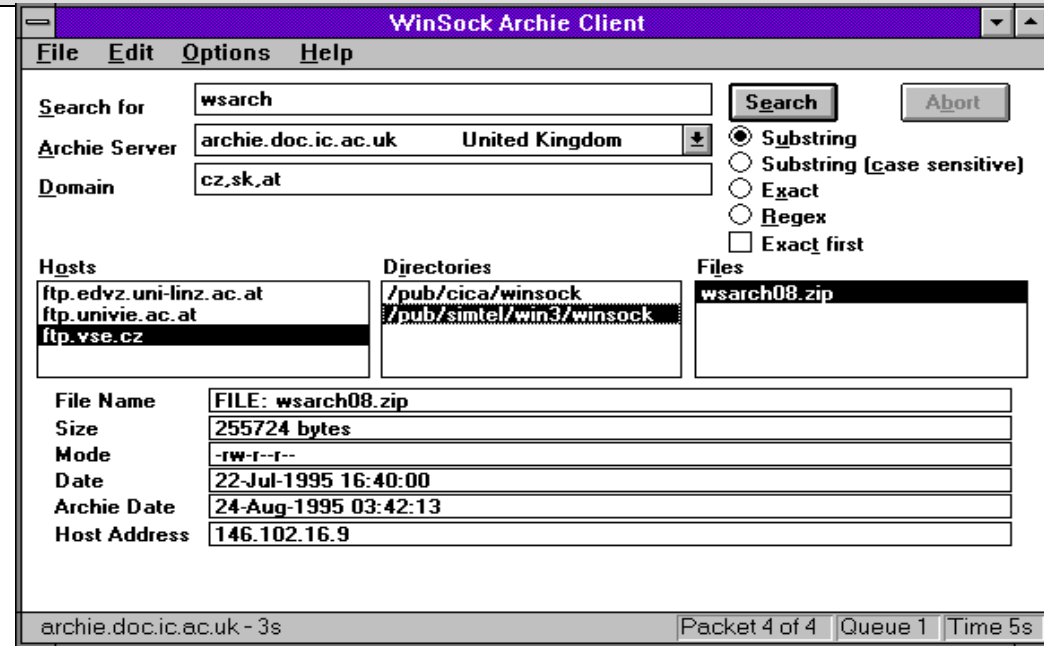
Gopher

- dnes již v Internetu funguje jen velmi málo serverů Gopher
 - např. `gopher://gopher.quux.org/`



specializované vs. nadstavbové služby v Internetu

- některé služby Internetu původně vznikly jako samostatné
 - byly pro ně vytvořeny samostatné (aplikační) protokoly a aplikace
 - klientské aplikace i servery
- například:
 - vyhledávání souborů v FTP archivech
 - služba Archie
 - plnotextové vyhledávání v dokumentech
 - služba WAIS (Wide Area Information System)
 -



WWW a el. pošta jako aplikační platformy

- původně samostatné služby (Archie, WAIS, ...) vyžadovaly, aby uživatelé:
 - používali specifické klientské aplikace
 - museli si je instalovat, konfigurovat atd.
 - používali specifický styl práce
 - učili se znát ovládání aplikací, příkazy atd.

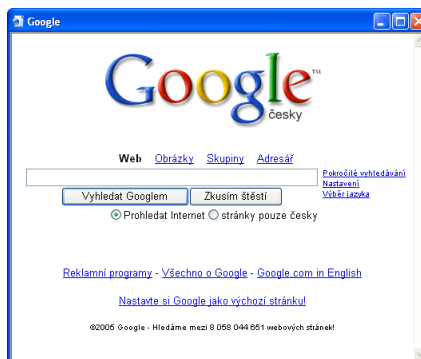
- celkový trend vedl k:

- minimalizaci klientů

- kvůli správě klientského SW
- kvůli nárokům na uživatele
-

- důsledek:

- původně široký repertoár služeb a aplikací v Internetu a TCP/IP se postupně zužoval
- až zůstaly dvě "základní aplikace", resp. služby, resp. klienti:
 - WWW (browser) a pošta (poštovní klient)



- elektronická pošta a WWW se staly platformami, na kterých jsou "stavěny" další aplikace

- takové, které původně byly samostatné

- elektronická pošta:

- zprostředkovává též: diskuse (News, NetNews, Usenet), elektronické konference, nástěnky (bulletin-board) apod.

- WWW:

- nejružnější formy vyhledávání
 - obecné i specializované
- transakce
 - objednávání, nakupování, prodej, ...
- hry, e-learning,
- vzdálené přihlašování

- přesto stále vznikají samostatní klienti

- např. pro instant messaging apod.