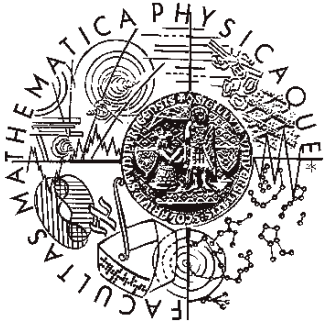
Počítačové sítě, v. 3.4



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

Lekce 12: Vývoj výpočetního modelu

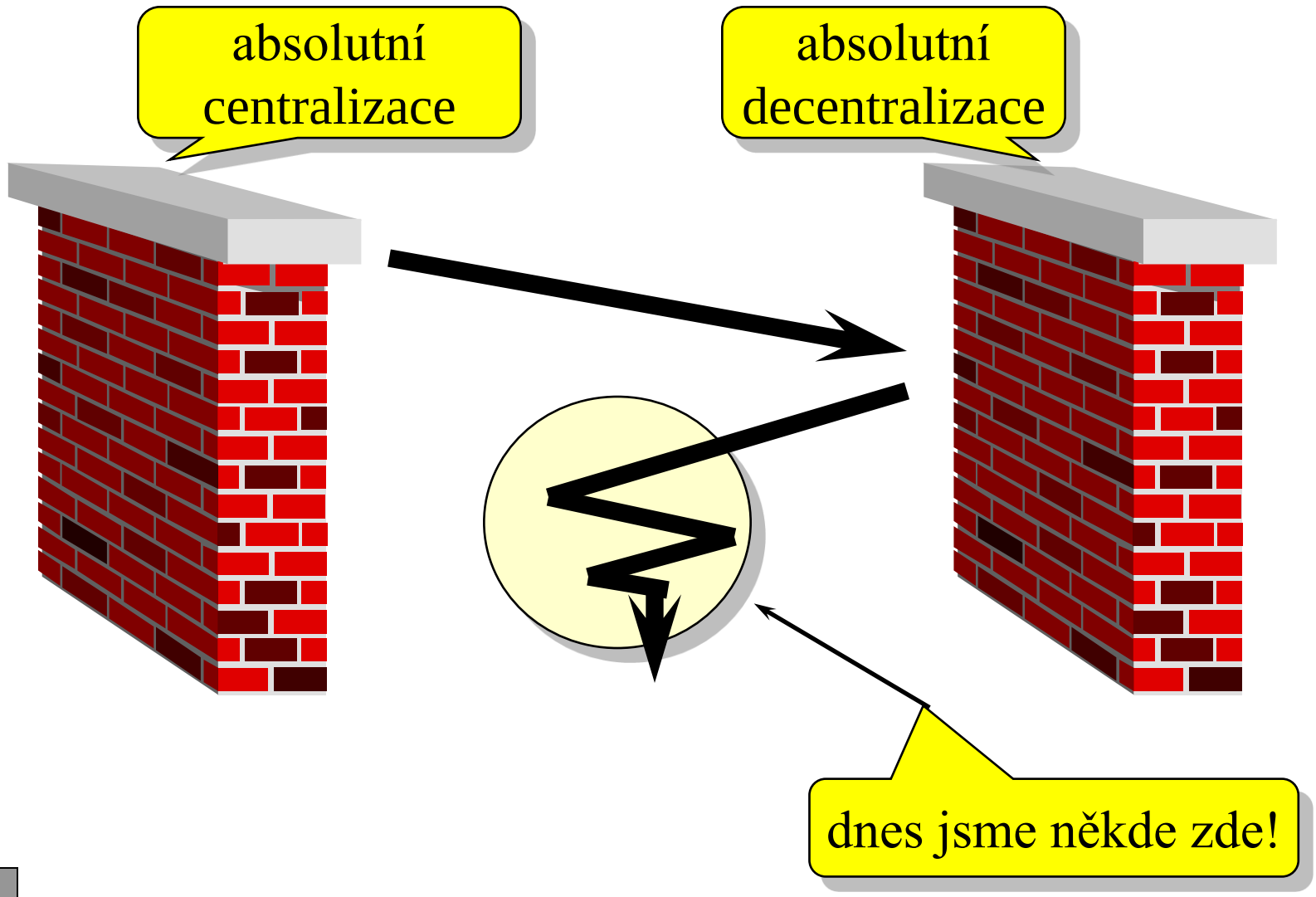
J. Peterka, 2009

co je výpočetní model?

- ucelená představa o tom,
  - kde jsou aplikace uchovávány jako programy a kde skutečně běží
  - zda (a jak) jsou aplikace rozděleny na části, jak tyto části vzájemně spolupracují
  - kde a jak se uchovávají a zpracovávají data
  - kde se nachází uživatel, kdy, jak a jakým způsobem komunikuje se svými aplikacemi
  - .....
- výpočetní model se vyvíjel a stále vyvíjí
  - některé výpočetní modely nepočítají s existencí sítě (např. dávkové zpracování)
  - jiné výpočetní modely spíše počítají s existencí sítě (např. klient/server)
  - další modely nutně vyžadují existenci sítě (např. distribuované zpracování, network-centric computing, thin-client, server-centric computing, on-demand computing, web services, …)

správné pochopení výpočetních modelů je důležité i pro zvládnutí problematiky sítí, pochopení jejich podstaty …

jak se vyvíjel výpočetní model?



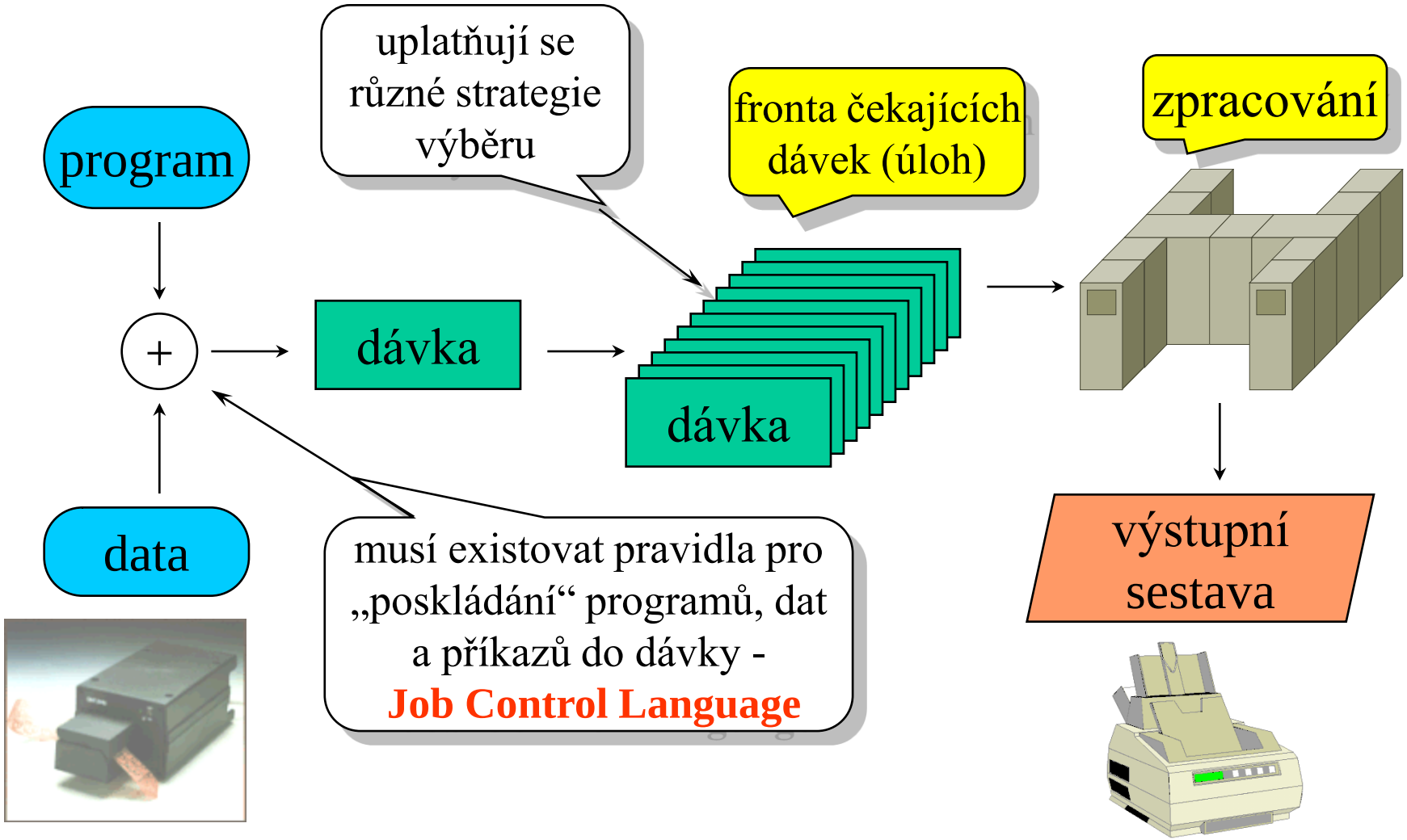
dávkové zpracování (batch processing)

- historicky nejstarší výpočetní model
- byl vynucen dobou
  - (ne)dokonalostí technologické základny
    - málo výkonný HW
  - malými schopnostmi SW
    - nebyla systémová podpora multitaskingu
  - vysokými náklady
  - potřebou „kolektivního“ využití dostupné výpočetní techniky
- dnes ještě není mrtvý!!!

tzv.“obrátka“,
trvala např. několik hodin až dní

- princip:
  - zájemce si dopředu připravil celý svůj „výpočet“
    - program
    - vstupní data
    - pokyny pro zpracování
  - a vše „zabalil“ do jednoho celku
    - tzv. dávky (angl: job)
    - např. v podobě sady děrných štítků či svitku děrné pásky
  - dávka se (fyzicky) přenesla k počítači a zařadila do fronty čekajících dávek
  - když na ni přišla řada, dávka se zpracovala
  - vznikl výstup (např. tisk)
    - na který mohl autor dávky reagovat, například opravou chyby, změnou vstupních dat

podstata dávkového zpracování



vlastnosti dávkového zpracování

NEvýhody:

- uživatel nemá bezprostřední kontakt se svou úlohou
  - chybí interaktivita
  - uživatel nemůže reagovat na průběh výpočtu (volit varianty dalšího průběhu, opravovat chyby, ...)
- doba obrátky bývá relativně dlouhá

Výhody:

- dokáže (relativně) dobře vytížit dostupné zdroje
  - vychází vstříc intenzivním výpočtům (hodně „počítavým“ úlohám, s minimem V/V)
- nutí programátory programovat „hlavou“ a ne „rukama“
  - protože při dlouhé obrátce si nemohou dovolit experimentovat)

Později:

- v prostředí sítě se používalo tzv. vzdálené zpracování úloh (Remote Job Execution, Remote Job Entry):
  - uživatel na jednom uzlu připravil dávku
  - poslal ji ke zpracování na jiný uzel
  - !! uživatel sám určoval, kam dávku pošle!!!

Dnes:

- modernější alternativa RJE *(„ ... distribuovaná aplikační platforma ... “??)*
  - síť si sama určuje, kam pošle dávku ke zpracování

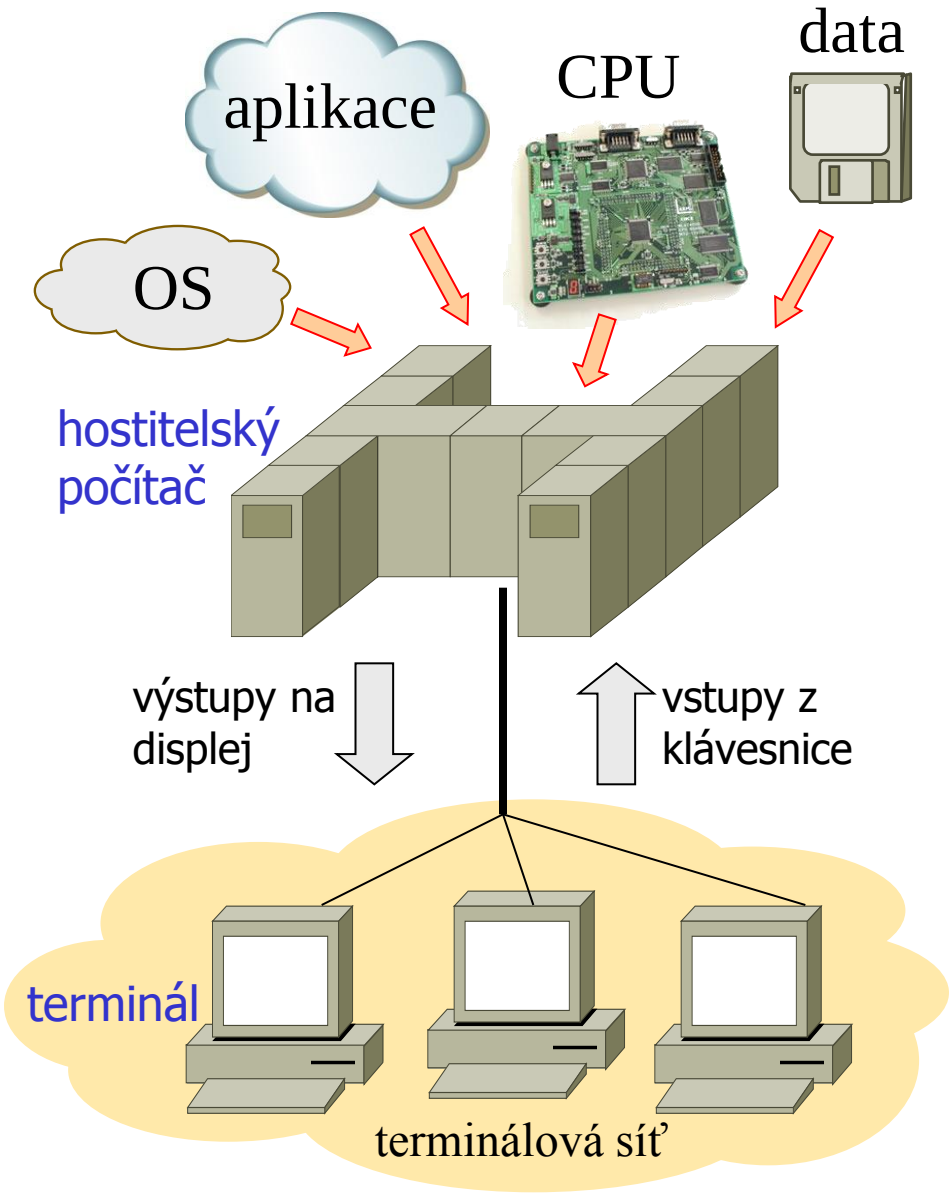
Do budoucna:

- model autonomních agentů
  - samostatní agenti (programy) dostanou určité zadání a to v prostředí sítě plní (samostatně, autonomně)

výpočetní model host/terminál

- vznikl jako reakce na neinteraktivnost dávkového zpracování
  - dokáže uživatelům zajistit přímý kontakt s jejich úlohami a interaktivní způsob práce
  - dokáže „obsloužit“ více uživatelů současně
- byl umožněn zdokonalením SW a HW:
  - SW mechanismy pro sdílení času (time sharing)
  - existencí uživatelských pracovišť (terminálů)
- host = počítač, který je „hostitelem“ systémových zdrojů
  - procesoru, paměti, V/V zařízení
  - programů, dat, systémových utilit, ...

odsud: hostitelský počítač (host)



podstata modelu host/terminál

- vše je „*na jedné hromadě*“
  - programy (úlohy) běží na hostitelském počítači
  - data se zpracovávají v místě kde se nachází (nedochází k přenosům velkých objemů dat)
- mezi hostitelským počítačem a terminály se přenáší pouze:
  - výstupy na obrazovku uživatele
  - vstupy z uživatelovy klávesnice
- terminály mohou být umístěny v různé vzdálenosti
  - blízko (místní, lokální terminály)
  - daleko (vzdálené terminály)
  - ..... (kdekoli v síti)
- „model host/terminál“ je způsob fungování
  - tj. „hostitelský počítač“ je role, ve které nějaký konkrétní počítač vystupuje
  - „střediskový počítač“, „mainframe“ atd. jsou kategorie (typy, třídy) počítačů
- mainframe může fungovat:
  - dávkově (používat dávkové zpracování)
  - jako hostitelský počítač (v režimu sdílení času)
- jako hostitelský počítač může fungovat např. PC s Unixem
  - rozhodující je charakter OS!!!

vlastnosti modelu host/terminál

Výhody:

- má centralizovaný charakter
  - správu stačí zajišťovat na jednom místě
  - snazší sdílení dat, programů, ...
- relativně snadná implementace
  - neklade příliš velké nároky na aplikace
- neklade velké nároky na přenos dat mezi hostitelským počítačem a terminály
  - přenáší se pouze výstupy na obrazovku uživatele a vstupy z uživatelovy klávesnice
    - *!!jsou to malé objemy dat, protože se (typicky) pracuje ve znakovém režimu!!*

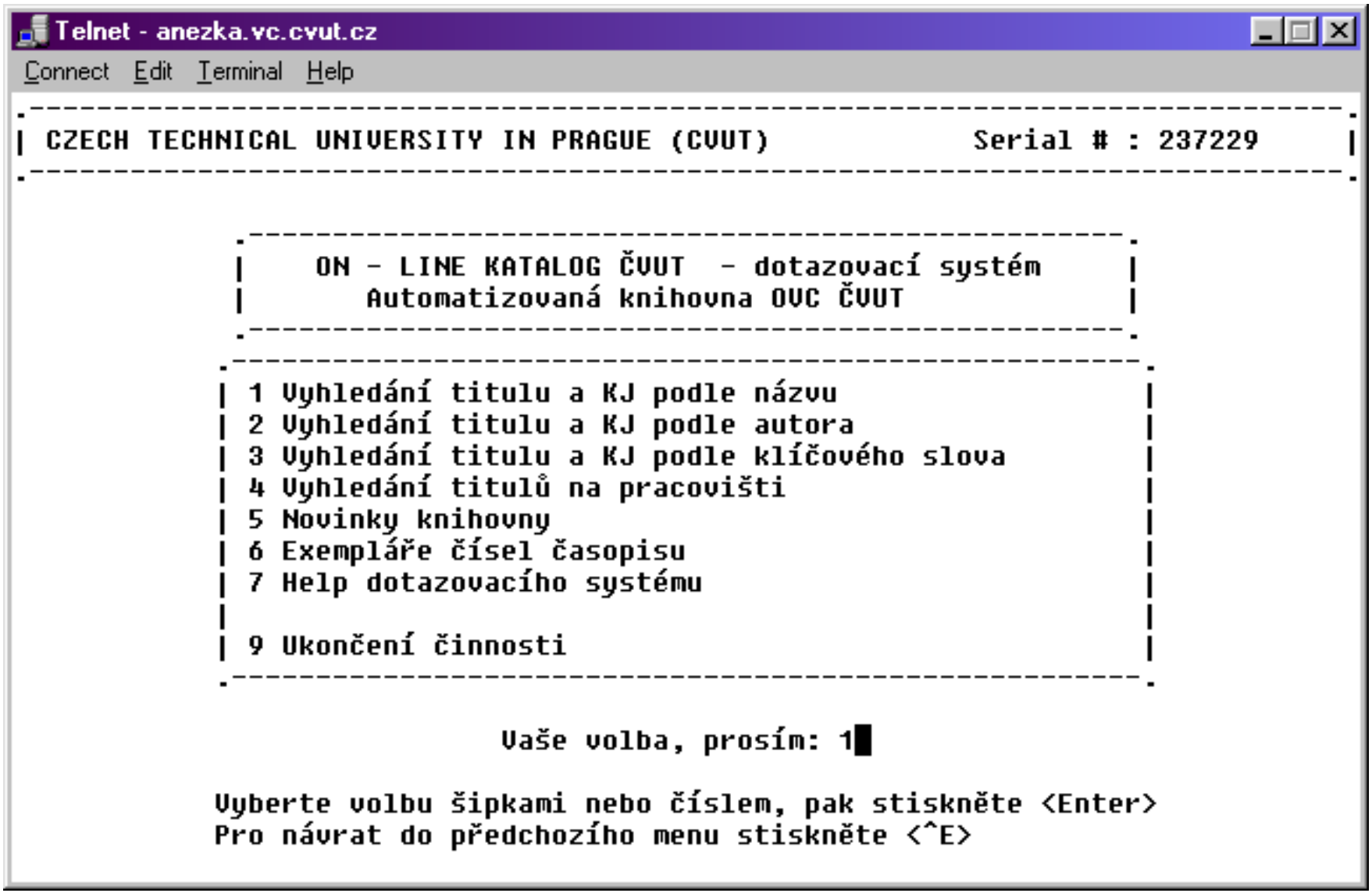
NEvýhody:

- uživatel má iluzi, že má hostitelský počítač výhradně ke své dispozici
  - ale ve skutečnosti má k dispozici jen n-tou část jeho výkonnosti!
- uživatelský komfort je relativně nízký
  - vzhledem ke znakovému režimu

!!! není to vina výpočetního modelu, ale způsobu jeho využití!!!

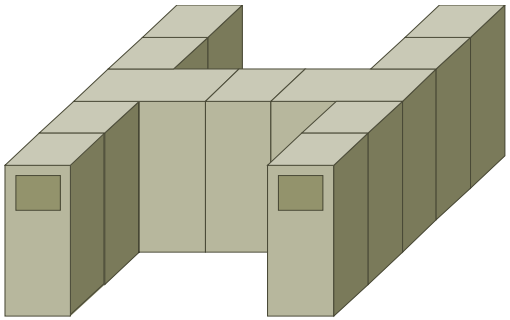
dnes již existuje možnost terminálového přístupu v grafickém režimu !!!

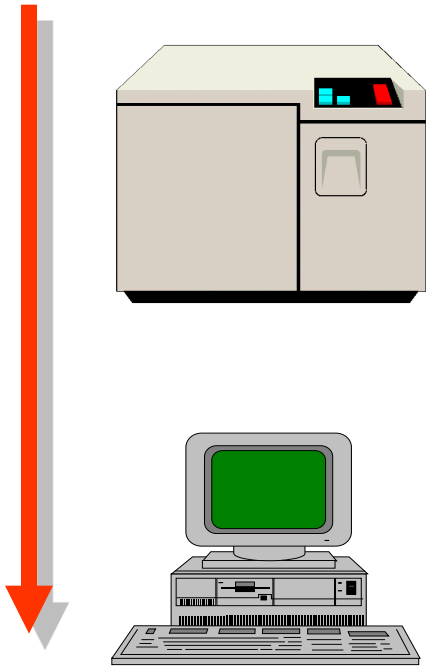
příklad
(aplikace provozovaná v režimu host/terminál)

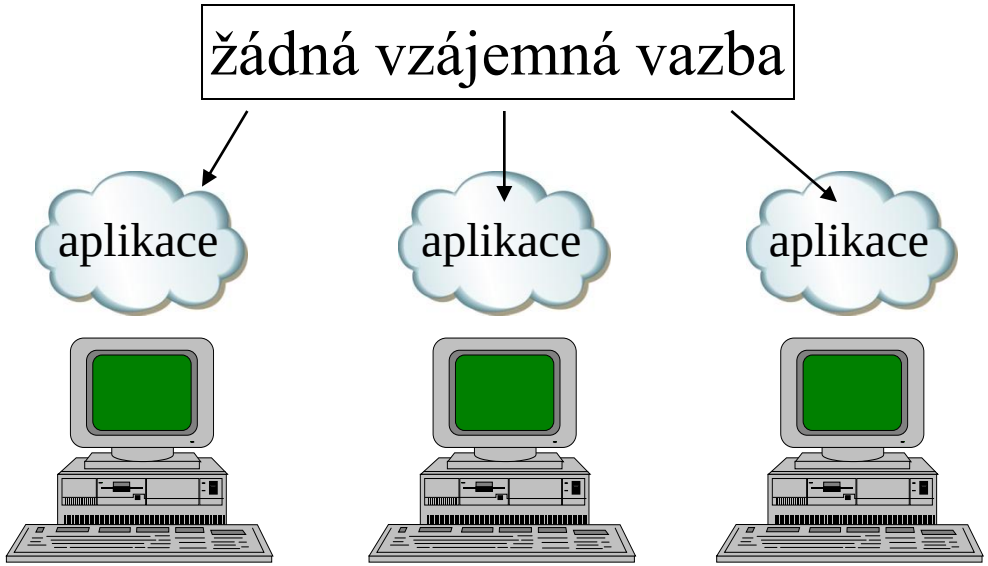


další vývoj: osobní počítače

- výpočetní technika se postupně stávala čím dál tím lacinější
  - zrodily se minipočítače
  - ale výpočetní model se nezměnil!!!!
- pořád bylo nutné (z ekonomických důvodů), aby více uživatelů sdílelo jeden počítač
- zlom nastal až s příchodem osobních počítačů
  - kdy už bylo ekonomicky únosné přidělit každému uživateli jeho vlastní počítač, k výhradnímu použití



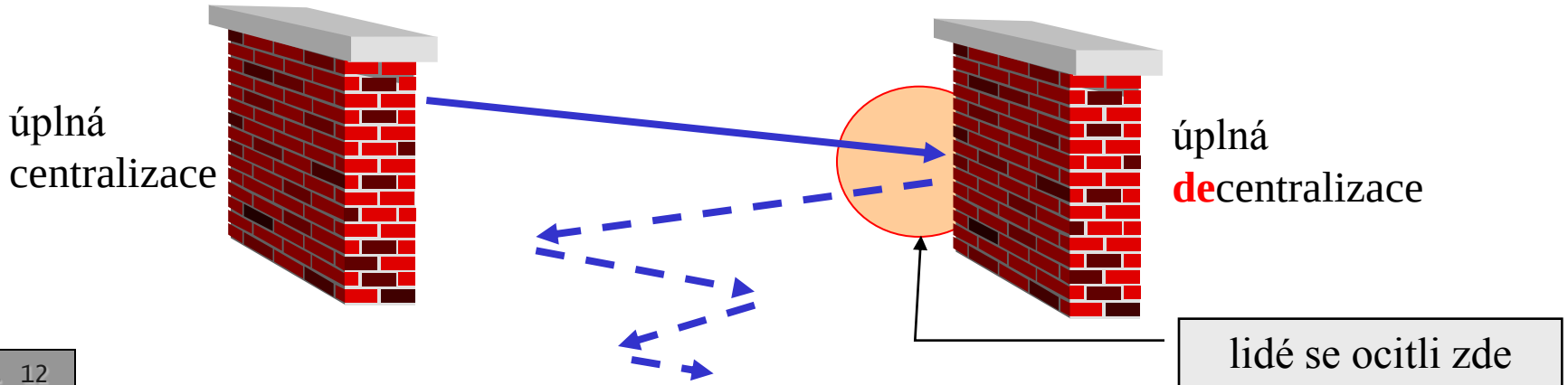




éra izolovaných počítačů

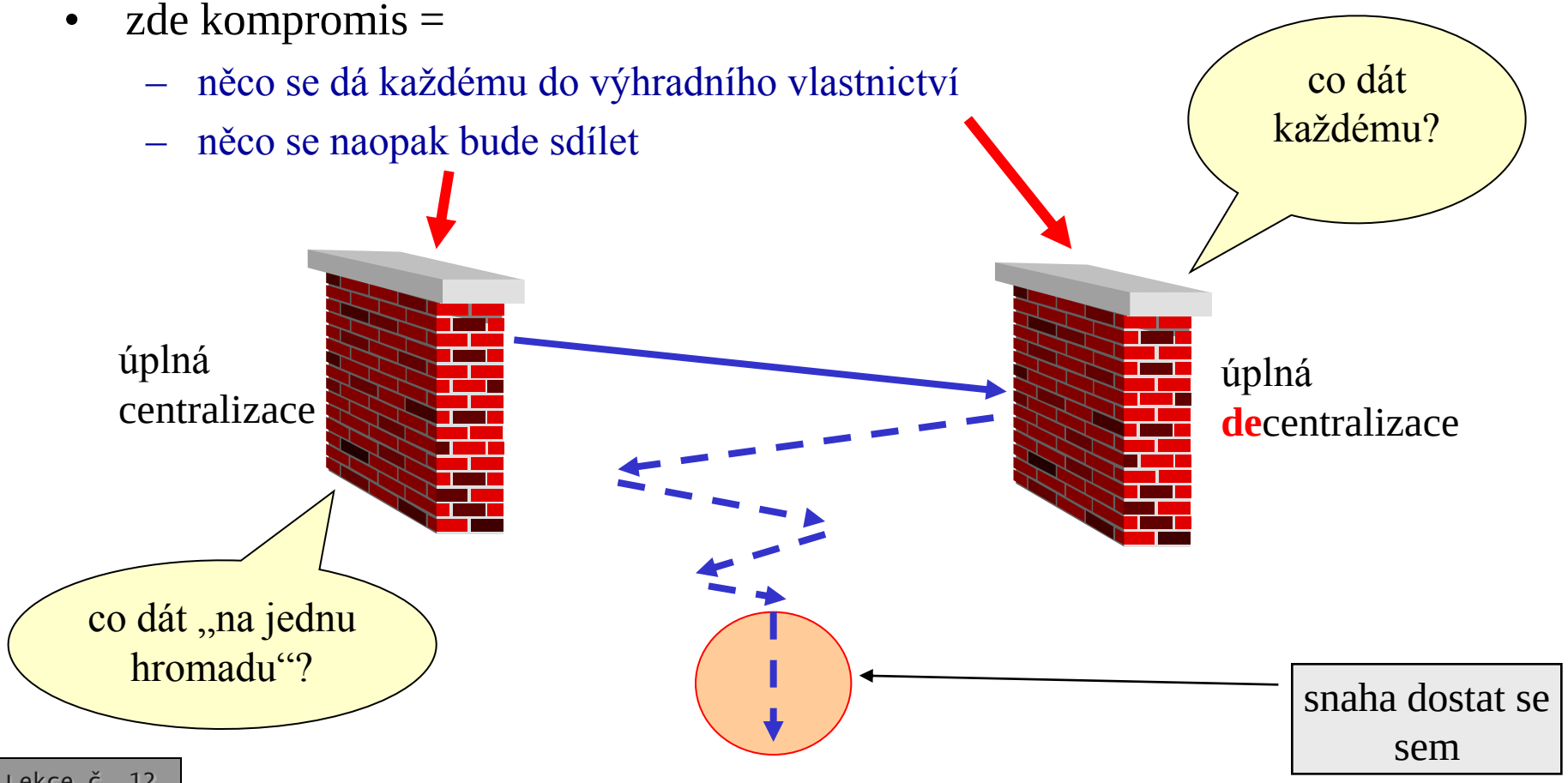
- od příchodu osobních počítačů si lidé slibovali především:
  - vyšší komfort
  - větší pružnost a flexibilitu
  - nezávislost na ostatních (žádnou potřebu sdílení)
- tyto požadavky se v zásadě podařilo splnit
- ale objevily se jiné problémy!!!

- dříve se každý problém řešil jednou, na jednom místě
  - nyní se každý problém řeší n-krát na n-místech
- uživatelé jsou mnohem více odkázáni na sebe
- jsou problémy se sdílením dat a programů
  - jak např. řešit práci nad společnými daty?
- některé věci (např. drahé periferie) není stále ještě únosné přidělit každému do výhradního vlastnictví



řešení: rozumný kompromis

- přísná centralizace (model host/terminál) i izolované osobní počítače jsou dva extrémy
- v životě většinou vítězí rozumný kompromis
- zde kompromis =
  - něco se dá každému do výhradního vlastnictví
  - něco se naopak bude sdílet



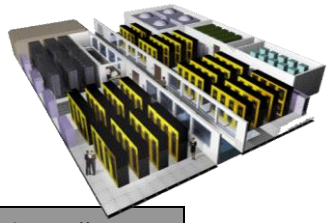
co má smysl …… ?

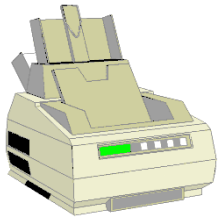
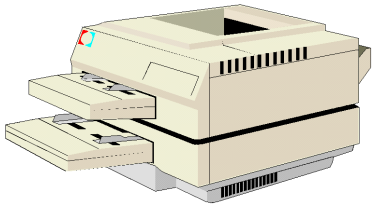
dát každému:

- vlastní výpočetní kapacitu
  - už je relativně laciná
- vlastní pracovní místo
  - klávesnici, monitor, myš, ...
  - uživateli lze vytvořit příjemné pracovní prostředí
- některé programy a data
  - nutno posuzovat individuálně

sdílet:

- drahé periferie
  - např. laserové tiskárny, modemy, ...
- společná data
  - firemní databáze, sdílené dokumenty, ...
- „soukromá“ data
  - např. kvůli zálohování
- aplikace
  - vyžadující správné nakonfigurování a „údržbu“



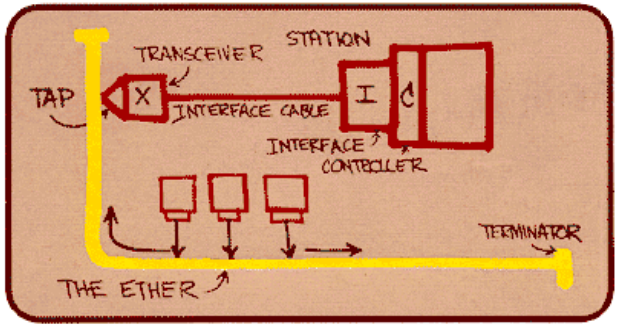



?

vznik prvních sítí LAN

- řeší především potřebu sdílení
  - souborů (programů, dat)
  - periferií (tiskáren, ...)
- uživatel nesmí sdílení poznat
  - uživatel nesmí pozorovat významnější rozdíl v rychlostech přístupu ke sdíleným a privátním objektům
  - je vhodné, když si uživatel vůbec nemusí uvědomovat fakt sdílení
  - mechanismy sdílení musí být implementovány transparentně
- jsou nutné dostatečně rychlé přenosové technologie
  - k dispozici je např. 10 Mbps Ethernet

- vše je realizováno jako lokální síť
  - LAN, Local Area Network
- sítě LAN jsou řešeny tak, aby je „*nebylo vidět*“
  - aby na nich mohly pracovat aplikace, které nejsou uzpůsobeny síťovému prostředí (neuvědomují si existenci sítě)
- teprve později se sítě mohou stát „viditelné“
  - když se objevují aplikace, které přímo počítají s existencí sítě



odbočení: vznik prvních sítí WAN

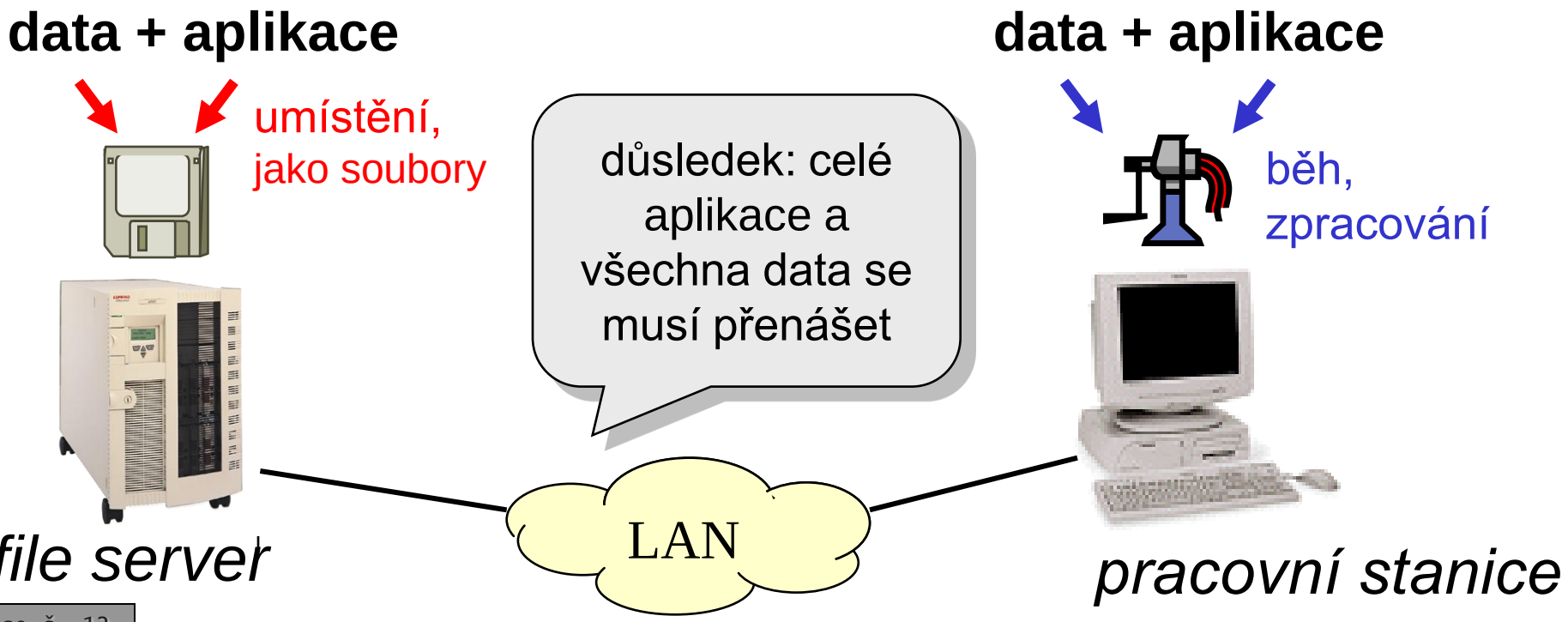
- jejich vznik je motivován spíše potřebou překlenout vzdálenost:
  - pro potřeby komunikace
  - pro potřeby sdílení výpočetní kapacity
  - pro potřeby sdílení dat
  - pro potřeby vzdáleného přístupu
  - .....
- vznikají první rozlehlé sítě
  - WAN (Wide Area Network)

- kvůli omezeným přenosovým možnostem (pomalým přenosům) na nich nelze dosáhnout transparentního sdílení
  - proto případné sdílení je řešeno netransparentně
    - uživatelé si uvědomují rozdíl mezi „místním“ a „vzdáleným“

přestává platit až se zaváděním broadbandu

nový model: file server / pracovní stanice

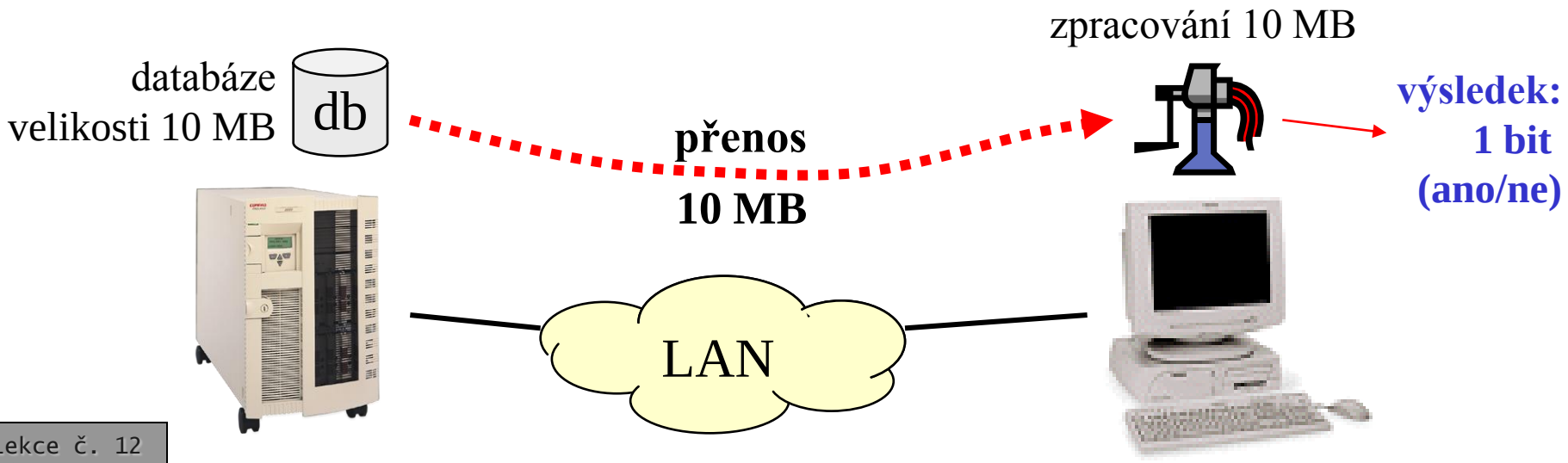
- nový výpočetní model pro sítě LAN
- snaží se vycházet vstříc potřebám sdílení v sítích LAN
  - aplikace a data jsou umístěna centrálně
    - na tzv. file serveru (souborovém serveru, jako soubory)
  - aplikace a data se zpracovávají (spouští) „lokálně“, na pracovních stanicích



model file server/pracovní stanice

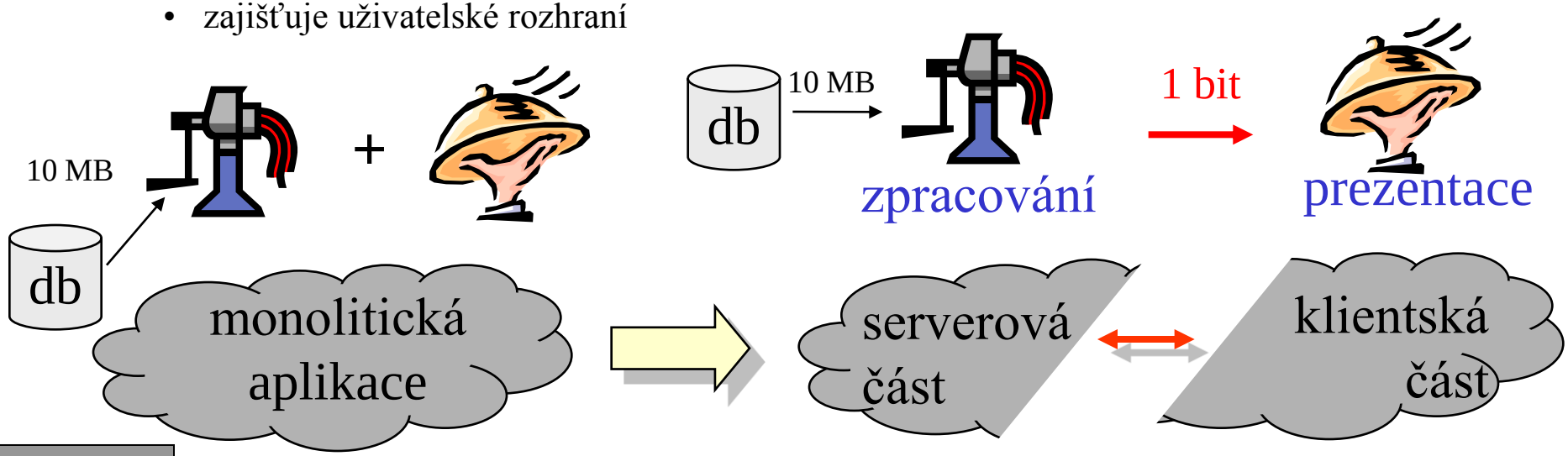
- pro aplikace je „neviditelný“
  - zajišťuje plně transparentní sdílení
- je použitelný i pro aplikace, které si neuvědomují existenci sítě
  - pro aplikace určené původně pro prostředí izolovaných počítačů
- umožňuje sdílení dat i programů
- umožňuje centrální správu

- v některých situacích je hodně neefektivní
  - způsobuje zbytečný přenos
  - může snadno dojít k zahlcení sítě
- důvod:
  - data jsou zpracována jinde, než jsou umístěna (a proto musí být přenášena)
  - podobně pro programy



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 sítích
- klient a server mohou stát na různých platformách



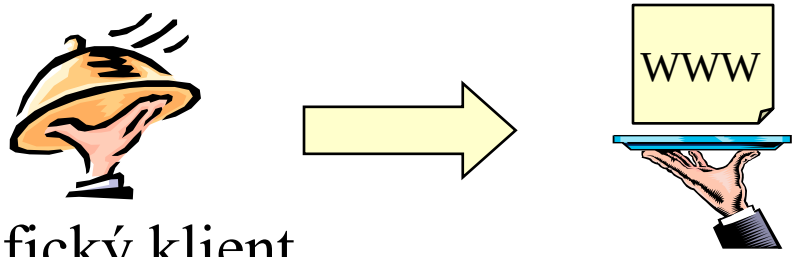
představa modelu klient/server

- komunikace mezi klientem a serverem se odehrává stylem: požadavek/odpověď
  - server pasivně čeká, až dostane nějaký požadavek.
  - komunikaci iniciuje klient, zasláním požadavku
  - musí být definována vzájemná komunikace mezi klientem a serverem
    - komunikační protokol (např. HTTP)
- mnoho služeb dnes funguje na bázi modelu klient/server
  - příklad: WWW (WWW server, WWW klient alias browser, protokol HTTP)
  - příklad: email (mail server, mail klient, protokol SMTP+POP3/IMAP ….)

nevýhody modelu klient/server

- klient není univerzální!
  - pro různé aplikace je nutné mít jinou klientskou část
    - s jiným ovládáním, jiným nastavováním, jinou správou atd.
  - s vývojem aplikace dochází i k vývoji klientské části
    - uživatelé si musí instalovat a udržovat nové verze klientských programů
- způsobuje to značné problémy
  - se systémovou správou, s podporou uživatelů
    - s každou aplikací se pracuje jinak
- důsledek:
  - nárůst nákladů TCO (Total Cost of Ownership)

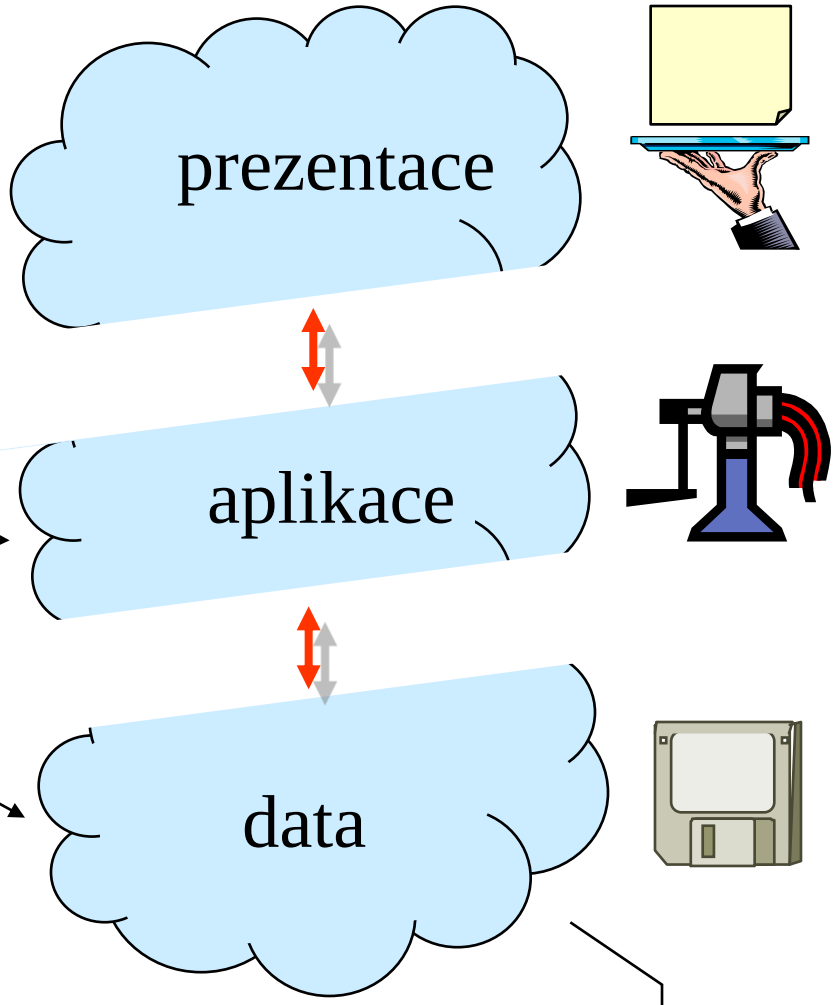
- možné řešení:
  - rozdělit aplikaci na 3 části
    - prezentační
    - aplikační
    - datovou
  - tak, aby se to, co je specifické pro danou aplikaci, soustředilo do „prostřední“ části
  - a aby se obě „krajní“ části nemusely měnit, resp. lišit pro různé aplikace
- přínos:
  - lze použít univerzálního klienta
    - současně, pro různé služby



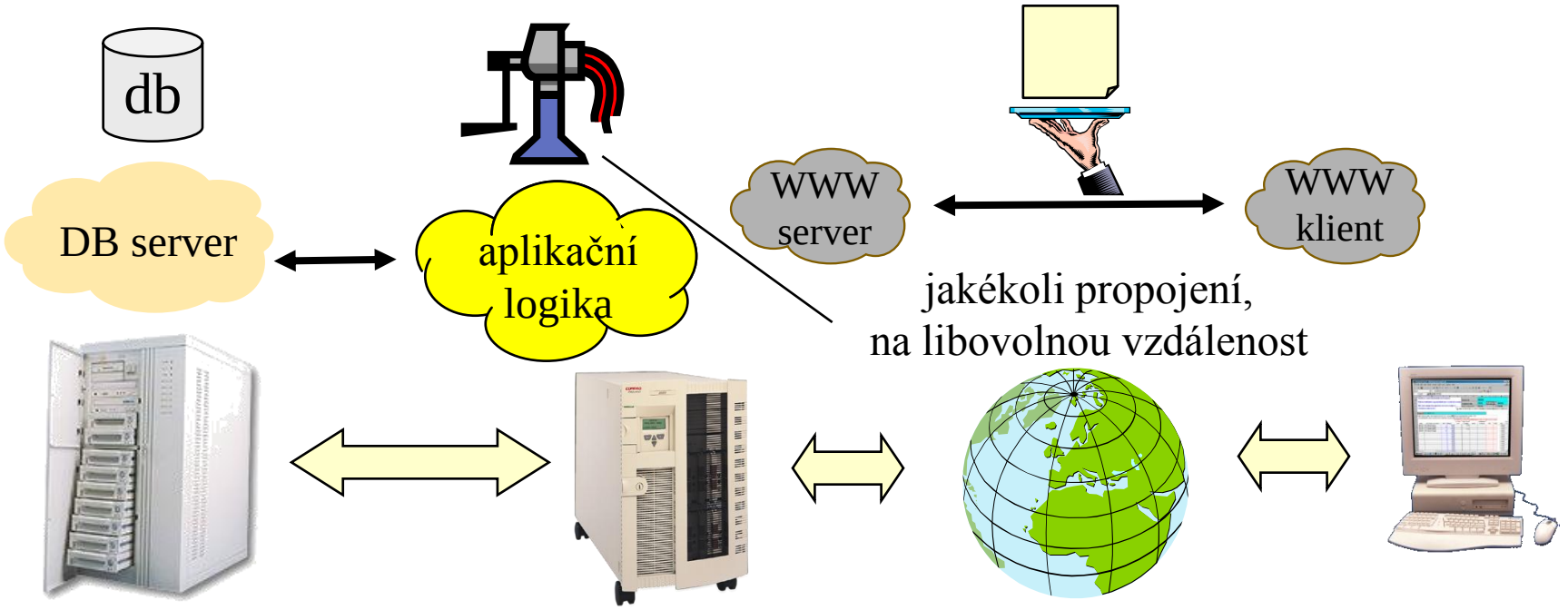
specifický klient

3-úrovňová architektura klient/server

- klasické řešení klient/server:
  - rozděluje aplikaci na dvě části
  - vzniká dvouvrstvá architektura
- novější řešení - rozdělení funkcí do 3 částí:
  - prezentační funkce
    - uživatelské rozhraní, sběr dotazů, prezentace výsledků
  - aplikační funkce
    - vlastní logika aplikace
  - správa dat
    - vlastní databázové operace
- lze implementovat jako:
  - 3 úrovňové řešení
  - 2 úrovňové řešení (celkem 5 možností)

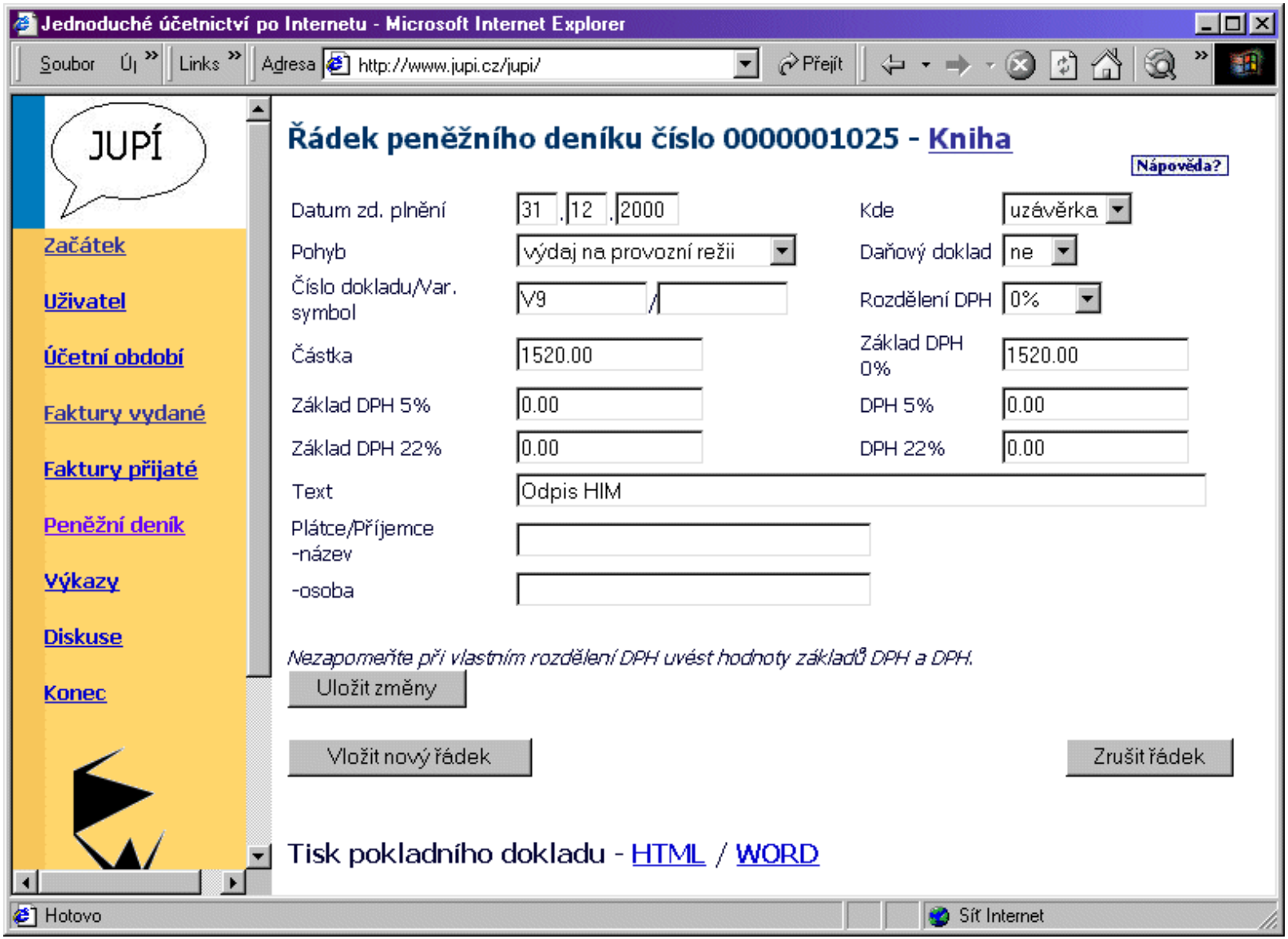


představa 3-úrovňové klient/server aplikace

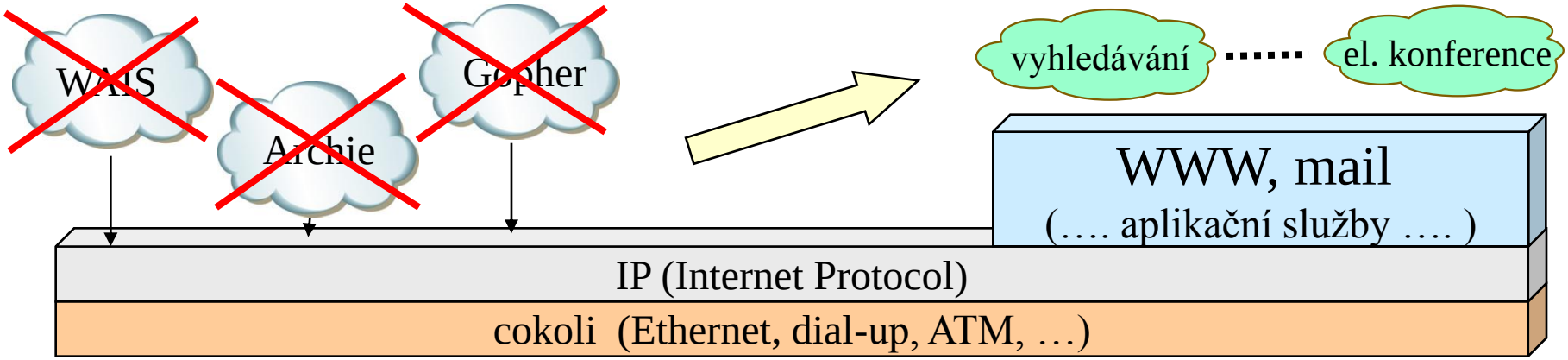


- výhody:
  - klient může být velmi univerzální (WWW browser)
    - a se změnami aplikace se nemusí měnit
    - uživatelé pracují s různými aplikacemi/službami jednotným způsobem
  - vše specifické je před uživateli „schováno“
  - WWW server (i DB server) se mohou nacházet kdekoli
    - vzdálenost ani umístění WWW a DB serveru nehrají (významnou) roli

příklad (webové) aplikace
(jednoduché účetnictví po Internetu)



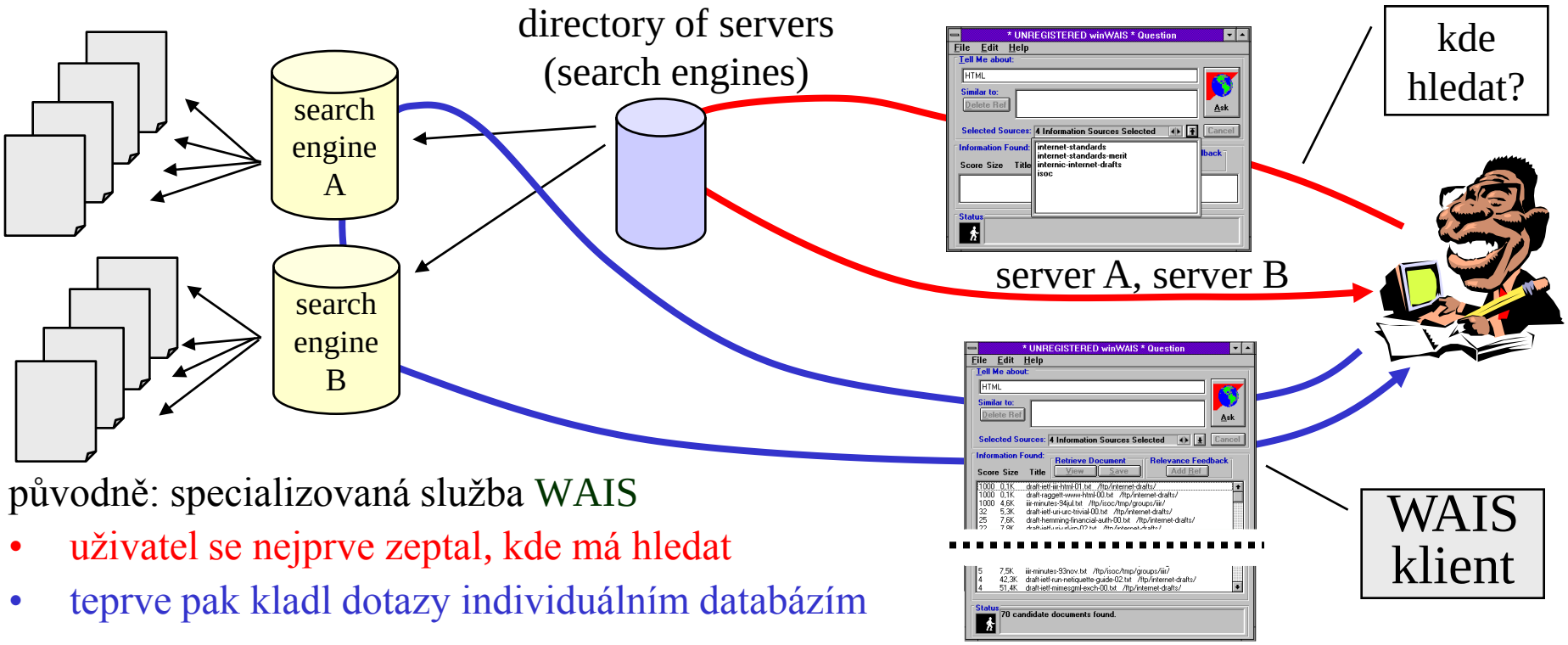
důsledky



- původně „samostatné“ (2-úrovňové klient/server) aplikace
  - s vlastními servery a klienty, vlastním stylem práce a ovládáním
- přechází do podoby „nesamostatných“ služeb, charakteru nadstavby nad WWW (event. el. poštu)
  - „schovávají se“ za WWW servery, uživatelé s nimi pracují skrze WWW
  - nemají vlastní klienty
    - jejich roli přebírají formuláře ve WWW
- příklady:
  - vyhledávání – původně samostatné aplikace, dnes skrze WWW
    - dříve: Archie, WAIS, Čmuchal atd., dnes Google, AltaVista, Jyxo …
  - informační (a další) on-line služby
    - např. Obchodní rejstřík, přímé bankovnictví (skrze WWW) atd.
  - webmail – práce s poštou skrze webové rozhraní
  - obecně: intranety a extranety místo „jednoúčelových“ aplikací

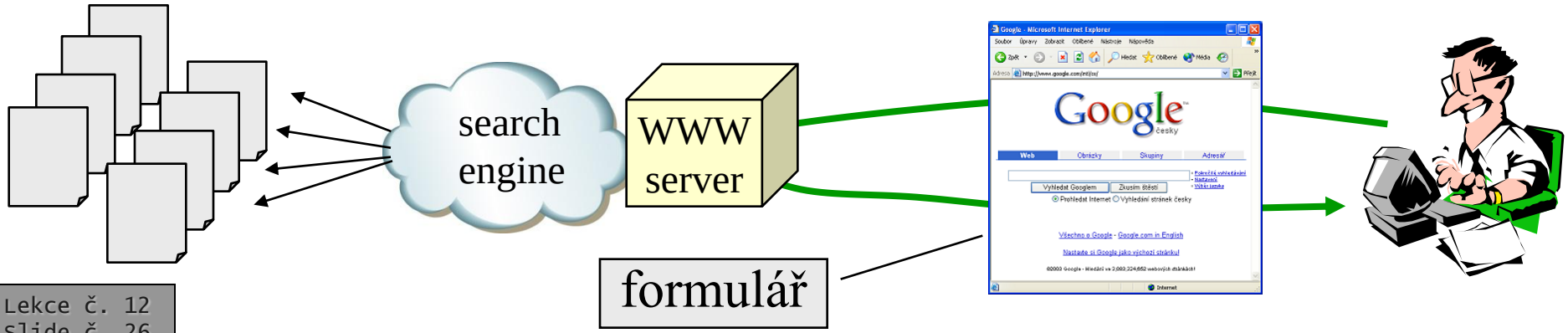
v zásadě přechází na 3-úrovňovou klient/server architekturu

příklad: plnotextové vyhledávání v Internetu



původně: specializovaná služba WAIS

- uživatel se nejprve zeptal, kde má hledat
- teprve pak kladl dotazy individuálním databázím

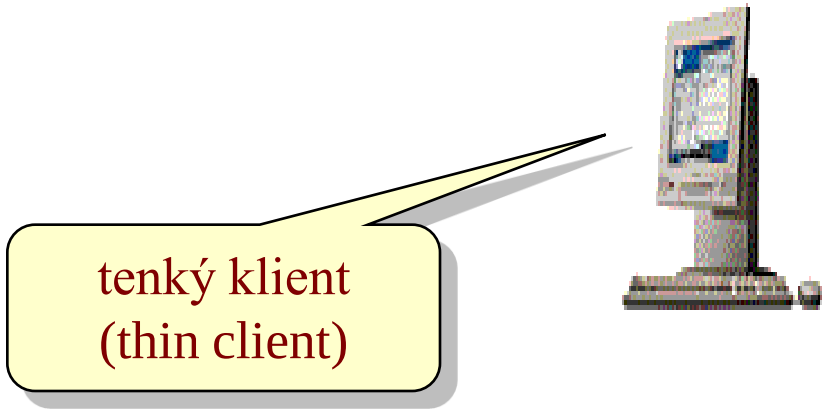


"Tlusté PC" vs. tenký klient

směr dalšího vývoje:

- snižovat náklady na provoz
  - v rámci TCO (Total Cost of Ownership)
- výchozí teze:
  - "klasické PC" musí být připraveno na vše, co by mohlo být zapotřebí
    - musí mít instalovány všechny programy které by uživatel mohl chtít použít
    - musí být podle toho dimenzováno (CPU, RAM, HD, …)
  - "klasické PC" je "tlusté"

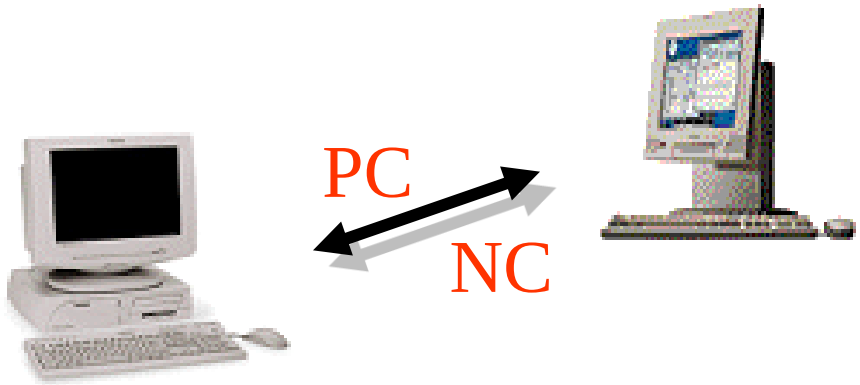
- návrh řešení:
  - neinstalovat programy dopředu, kvůli jejich POTENCIÁLNÍ potřebě
  - ale zavádět je až v okamžiku jejich AKTUÁLNÍ potřeby !!
- důsledek:
  - počítač (terminál, koncové zařízení, ….) stačí vybavit "minimalisticky", tím co potřebuje ke stažení (zavedení) toho co právě potřebuje
    - toto zařízení může být **"tenké"**



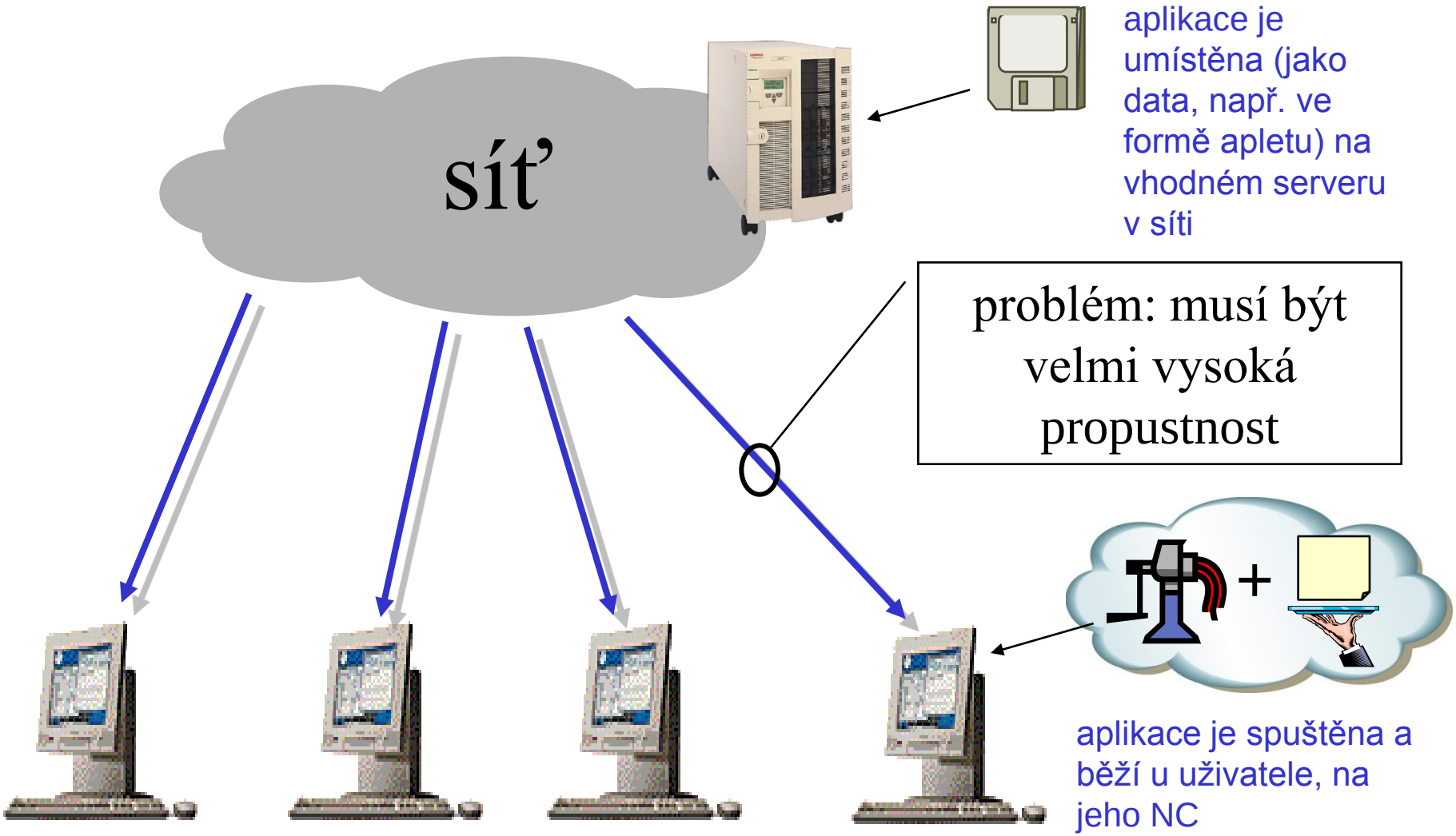
jak realizovat tenkého klienta?

- představa:
  - potřebné programy si tenký klient bude stahovat ze sítě
    - není až tak podstatné odkud,
    - výběr "zdroje" lze ponechat na "chytré síti" a jejím rozhodnutí
  - použitelným formátem jsou např. aplety jazyka Java
    - tenký klient pak musí být vybaven JVM (Java Virtual Machine)
    - jinak to může být maximálně jednoduchý stroj s nulovými nároky na systémovou správu!

- terminologie:
  - celému modelu fungování (výpočetnímu modelu) se začalo říkat "Network-Centric Computing"
    - protože síť se stává středem všeho, veškerá inteligence (i potřeba správy) je soustředěna do sítě)
  - pro "tenkého klienta" se vžil také název "Network Computer" (zkratkou NC)
    - jako určitý protipól PC alias "tlustého klienta"



představa fungování Network-Centric Computing



po „použití“ se aplikace jednoduše zahodí (vymaže z paměti NC)

osud tenkých klientů

- myšlenka tenkých klientů se v praxi příliš neujala
- důvodů bylo více:
  - nedostatečná kapacita sítě
    - nutná kvůli rychlé odezvě na aktivity uživatele
  - nepřipravenost aplikací a SW platformy …
    - **již existující aplikace nešlo použít !!!!**
    - snahy napsat celý kancelářský balík v Javě byly zastaveny
  - malý cenový rozdíl mezi NC a PC
    - ale velký ve funkčnosti
    - NC nedokáže pracovat samo při výpadku sítě, PC ano
  - "aktivní nezájem" odpůrců Javy
    - ……
- počítače NC však našly uplatnění
  - v rámci intranetů
    - kde je dostatečně dimenzovaná přenosová infrastruktura
  - pro specializované aplikace
    - kde mělo smysl vše napsat od základu znovu a ušít na míru potřebám uživatelů a prostředí NC
  - pro jednoúčelové nasazení
    - tam, kde uživatel používá NC stále pro jediný účel
      - např. pro nějakou agendu u přepážky
- *neúspěch NC se týká jejich nasazení pro "univerzální použití" v otevřenějším prostředí než je uzavřený intranet.*

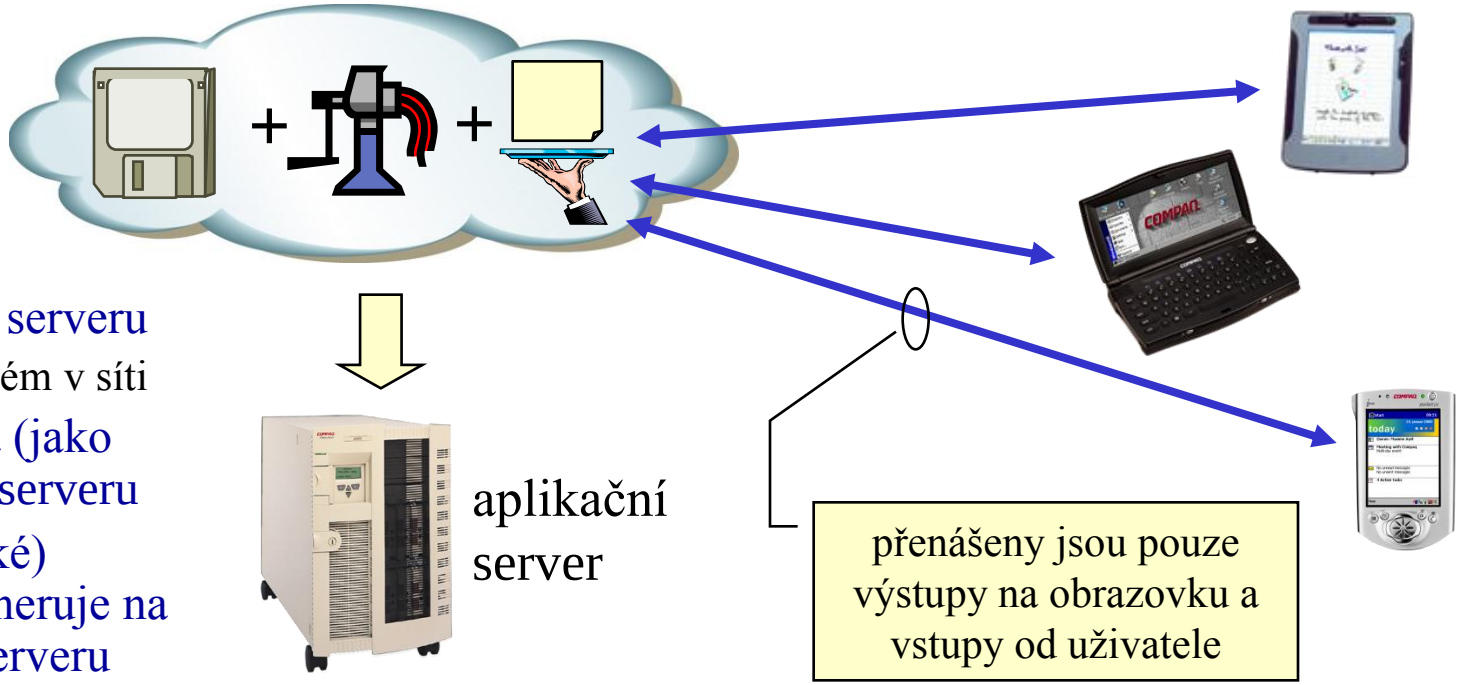
server-based computing
aneb: renesance modelu host/terminál

- cesta snižování TCO (nákladů na provoz) skrze NC se ukázala jako nepříliš schůdná
- další pokus se ubíral cestou návratu k plné centralizaci
  - návratu k modelu host/terminál
    - ale bez jeho problémů s nízkou uživatelskou přítulností
- další motivace:
  - snaha umožnit použití i jiných zařízení než jen PC

- technické předpoklady:
  - našla se řešení, která umožňují vzdálený terminálový přístup v grafickém režimu, při únosných nárocích na přenosovou kapacitu
    - X Window
    - Citrix ICA, MetaFrame, WinFrame
    - MS Terminal Server (ex Hydra)

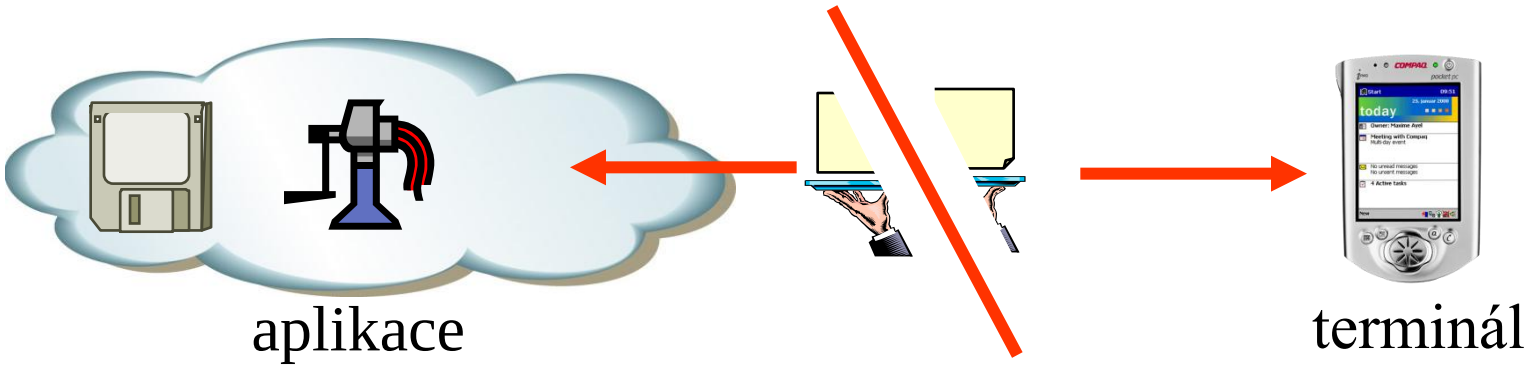
Server-Based Computing

- aplikace:
  - běží na tzv. aplikačním serveru
    - umístěném v síti
  - je umístěna (jako soubor) na serveru
  - své (grafické) výstupy generuje na aplikační serveru



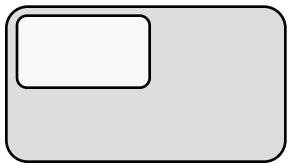
- v principu se jedná o návrat k původnímu modelu host/terminál
  - snahou je využít všech výhod centralizace ke snížení nákladů na provoz a správu (TCO)
  - ale bez ztráty komfortu pro uživatele (nutnost fungování v grafickém režimu)
- problém je v tom, že generovaná grafická data mohou být neúnosně velká, a vyžadovala by příliš velkou přenosovou kapacitu
  - je nutné jiné řešení, optimalizující objem přenášených dat

Server-Based Computing představa realizace

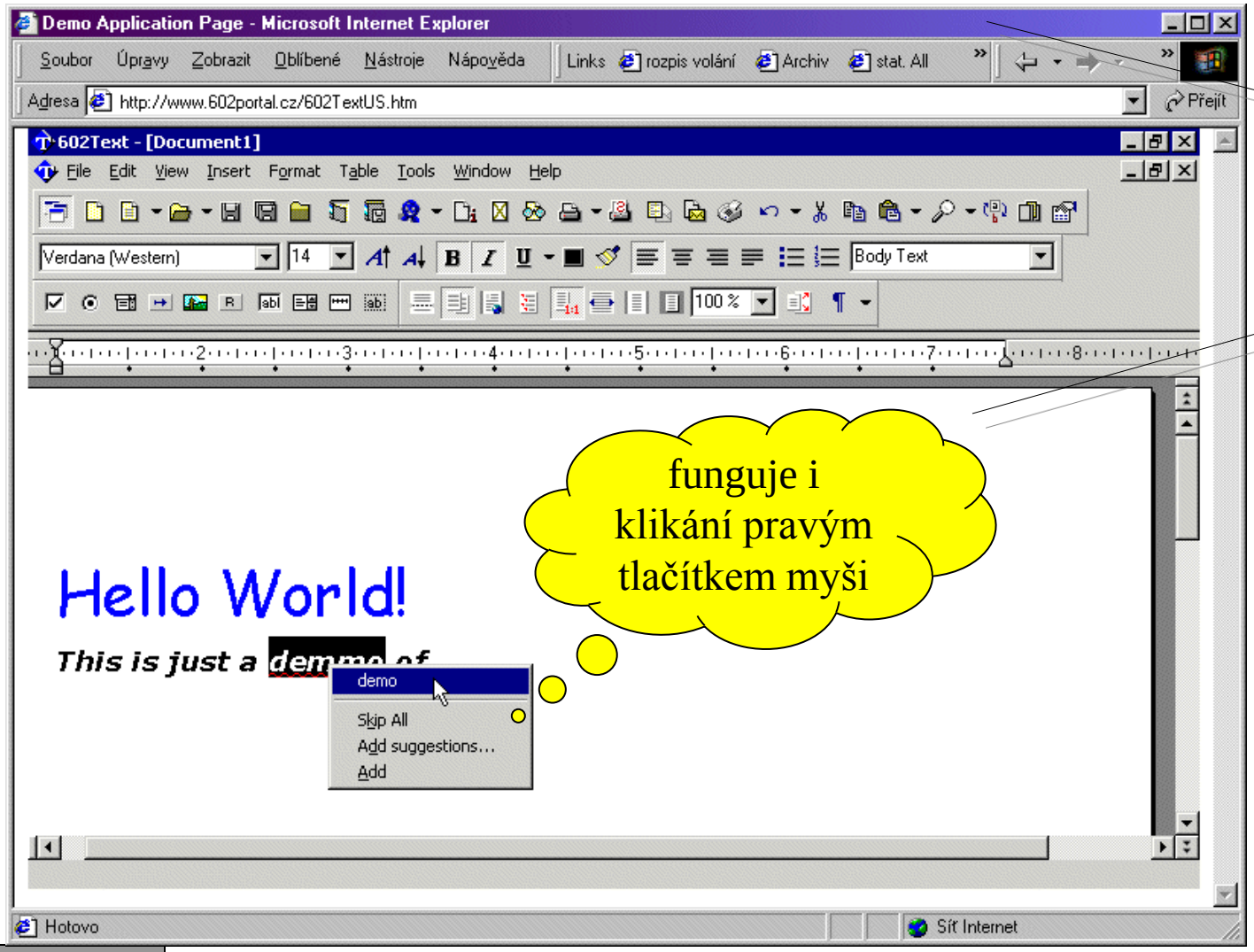


- řešením je vhodné „roztržení“ prezentačních funkcí
  - grafického subsystému („toho, co generuje grafická data“)
- a přemístění části generující grafická data přímo do terminálu
  - tak aby se objemná grafika generovala „místně“, a nemusela se nikam přenášet
  - lze se lépe přizpůsobit místním možnostem zobrazení
- „řez“ se musí udělat s ohledem na:
  - minimalizaci objemu přenášených dat
    - budou to příkazy (typu: vykresli okno“), nikoli přímo grafická (bitmapová) data
  - možnost implementace na platformě terminálu
- problém je s rozdílnými zobrazovacími schopnostmi různých terminálů
  - řeší se (částečně) pomocí tzv. panning-u
- příklady:
  - X Window, Citrix ICA, MetaFrame, WinFrame, MS Terminal Server

stačí např. i 9,6 kbps na 1 uživatele



Příklad: terminálový přístup skrze systém Citrix WinFrame (MetaFrame)



funguje i klikání pravým tlačítkem myši

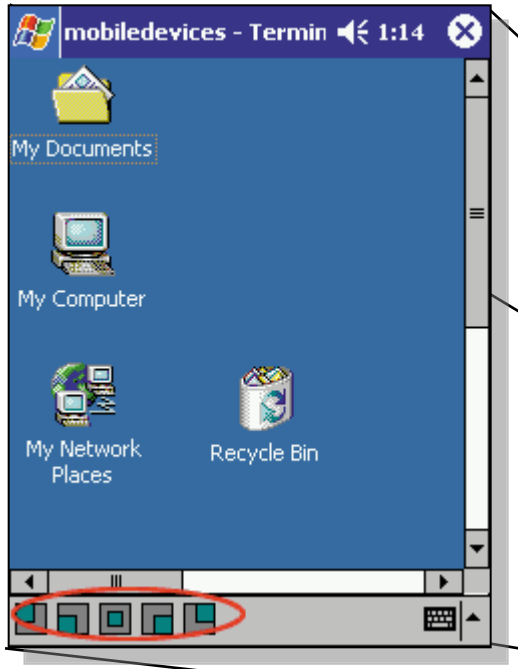
WWW browser

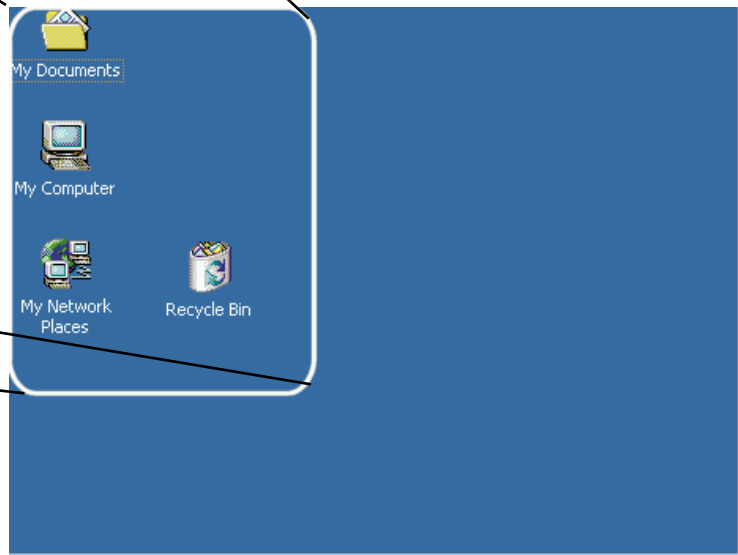
aplikace (textový editor), běžící na vzdáleném počítači

příklad: Terminal Services na PDA

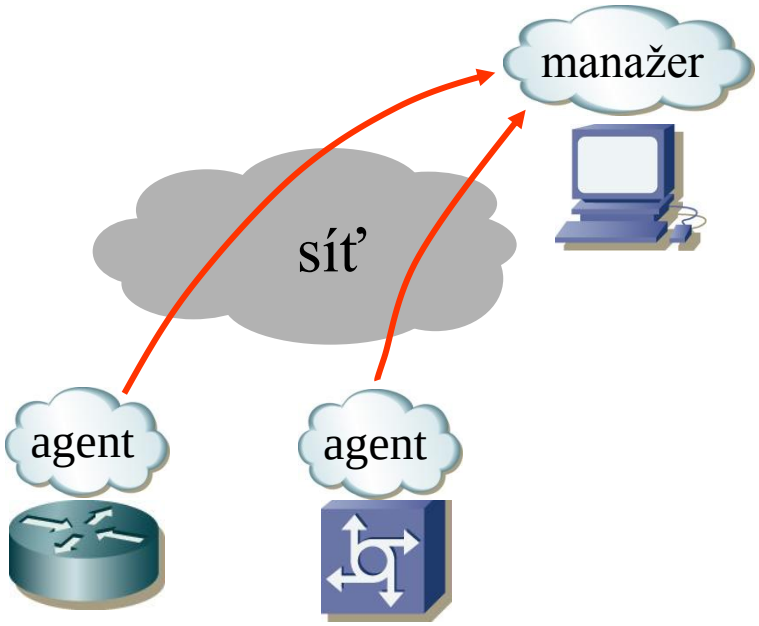
- lze se přihlásit ke vzdálenému "terminálovému serveru"
  - fakticky: aplikačnímu serveru
- a provozovat na něm aplikace

- disproporce mezi velikostí "virtuální pracovní plochy" a velikostí reálného displeje se řeší skrze tzv. panning
  - reálný display ukazuje jen výřez virtuální pracovní plochy





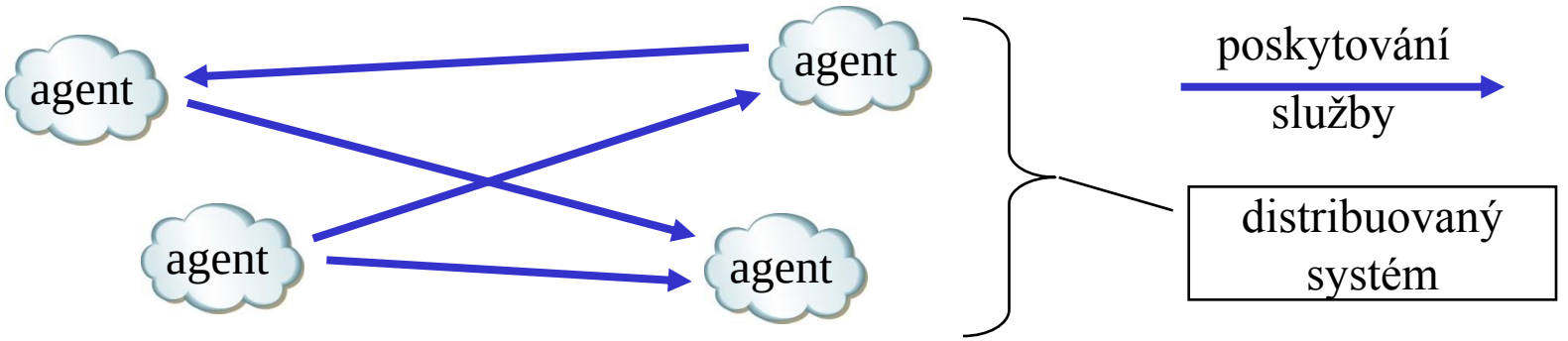
model agent/manažer



- agent:
  - „kus kódu“, který je někde umístěn, sbírá data/informace a posílá je do centra
- manažer:
  - je umístěn v centru, přijímá data od agentů a vyhodnocuje je

- původní využití:
  - pro management (správu)
    - agenti jsou zabudováni v různých zařízeních, monitorují jejich činnost, posílají zprávy o chybách a problémech do centra, manažerovi
    - manažer poskytuje přehled o stavu sítě
    - …
- perspektivně:
  - technologie tzv. inteligentních (a mobilních) agentů
    - agenti mají konkrétní zadání (např. hledat a sbírat informace), mají vysokou míru autonomie (mohou se samy rozhodovat co a jak dál), a při plnění zadaného úkolu se mohou také sami přemisťovat

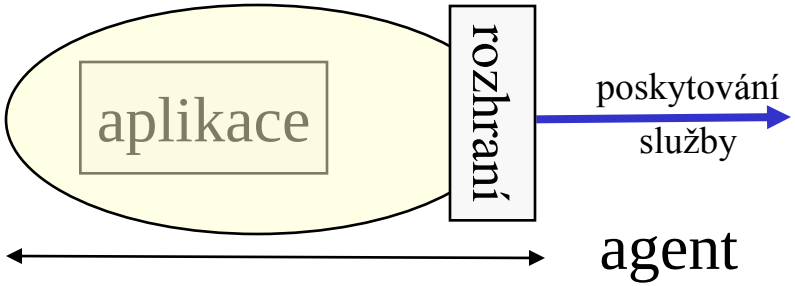
architektura orientovaná na služby



- dochází k úplnému oddělení
  - způsobu provozování a fungování aplikací
  - efektu, který to přináší (poskytované služby)
- obecně:
  - kdokoli (jakýkoli agent) může nabízet a poskytovat službu
  - kdokoli (jakýkoli agent) může využívat službu
- komunikace má charakter „požadavek/odpověď“
  - je bezestavová

je skryto

- musí být vyřešeno:
  - jak se agenti dozvědí o poskytovaných službách
    - vhodná adresářová služba, kde by byly uvedeny všechny poskytované služby
  - jak budou agenti vzájemně komunikovat
    - komunikační protokol – pro vznášení požadavků, vracení výsledků atd.
  - jak budou formulovány požadavky a odpovědi
    - jaký bude formát dat (požadavků, odpovědí …)

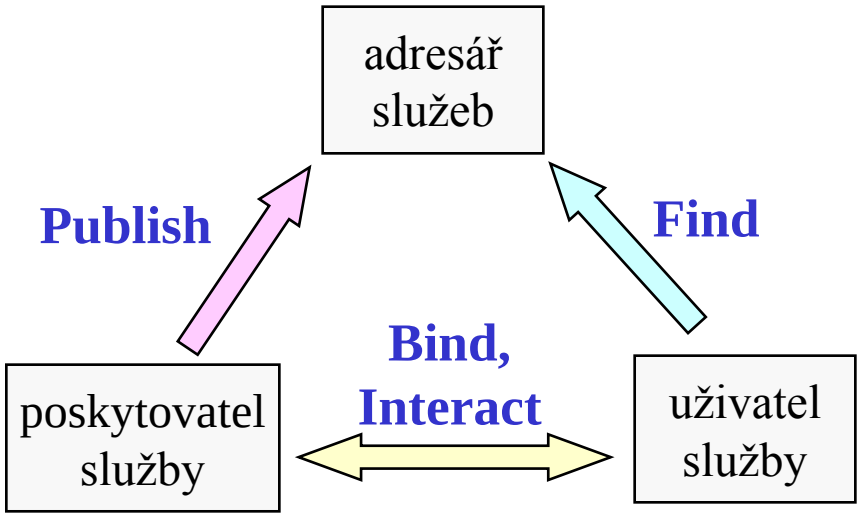


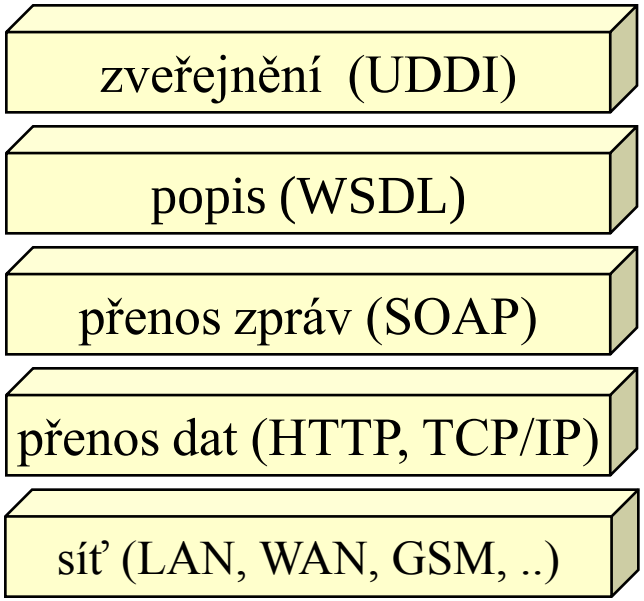
webové služby (Web Services)

princip:

- to co musí být zajištěno, je řešeno prostřednictvím technologií WWW
  - a nadstaveb nad WWW

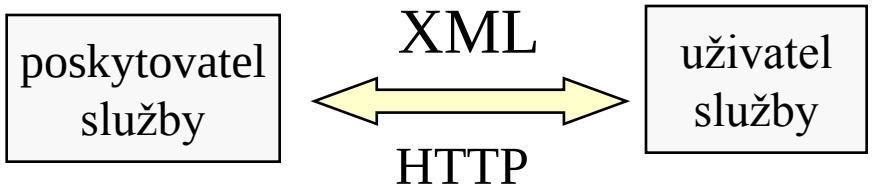
- UDDI
  - Universal Description, Discovery and Integration
    - pro zveřejnění popisu služby v rámci adresáře, pro vyhledávání služeb
- WSDL
  - Web Services Description Language
    - pro popis poskytovaných služeb
- SOAP
  - Simple Object Access Protocol
    - pro „zabalení“ požadavků a odpovědi do jednoho celku (zprávy), XML-based
- HTTP (a TCP/IP)
    - pro přenos dat





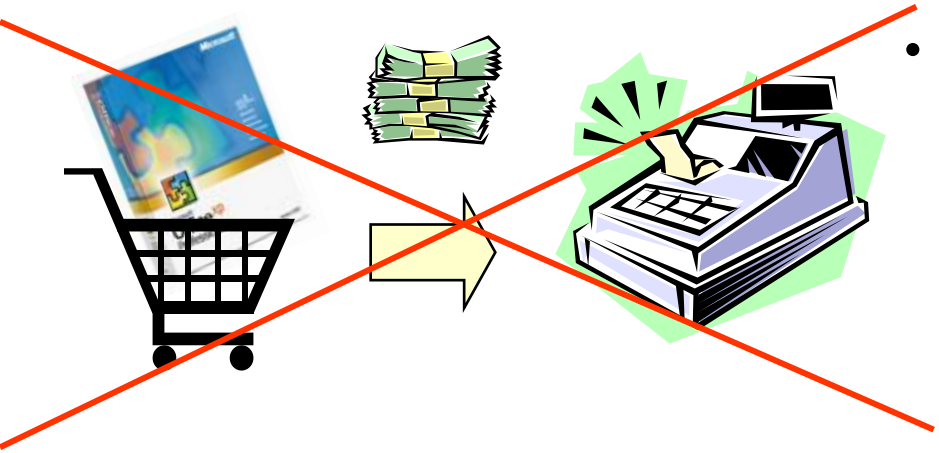
realita webových služeb

- webové služby jsou určeny pro vzájemnou komunikaci programů!!!
  - nepředpokládá se, že přímým uživatelem by byl člověk !!
    - jen přes další vrstvu vytvářející uživatelské rozhraní
- v praxi se webové služby využívají spíše „uvnitř“ firemních subjektů
  - pro jejich vnitrofiremní agendy a systémy
- nabídka webových služeb směrem „ven“ se rozjíždí velmi pomalu
  - příklad v ČR: objednávkový systém ADSL přípojek, provozuje Český Telecom



- pokud se dnes webové služby používají, pak stále ještě na „case-to-case“ bázi, bez existence adresářů webových služeb
  - tj. WDSL a UDDI se ještě moc nepoužívají
  - také použití SOAP je zatím nízké, spíše se používá přenos dat přímo v XML

SW jako služba



- tradiční přístup k SW:
  - uživatel si jej pořídí do svého vlastnictví (zakoupí), nainstaluje si ho, používá, stará se o něj …
  - struktura nákladů:
    - dobře predikovatelné jednorázové pořizovací náklady
    - špatně predikovatelné průběžné náklady na správu, podporu uživatelů, aktualizace atd.

nekupujte si SW, pronajměte si ho!

- princip **ASP** (Application Service Providing)
  - uživatel si SW nepořizuje do svého vlastnictví, neinstaluje si ho, neprovozuje ho
    - nemusí se o něj starat
  - uživatel SW pouze používá !!!
    - na dálku, prostřednictvím vzdáleného přístupu
      - na bázi server-based computing, či network-centric computing, či jako nadstavbovou službu nad WWW
  - aplikaci si pořizuje do svého vlastnictví subjekt ASP
    - poskytovatel aplikačních služeb (**ASP**, Application Service Provider)
    - stará se o provoz svého SW
    - prodává svému zákazníkovi použití tohoto SW
      - jako službu !!!

podstata ASP

http://www.sluzbyasp.cz

- nejde ani tak o nový výpočetní model, jako o "*ekonomický model*"
  - dochází k oddělení vlastníka od uživatele
    - dříve splývali
  - vlastník si pořizuje SW, stará se o něj, nese náklady na provoz (TCO), aktualizuje ….
    - jeho náklady jsou proměnlivé
    - nese riziko neúspěchu, nefunkčnosti
  - uživatel pouze používá funkce
    - odpadají mu počáteční pořizovací náklady
    - uživatel platí např. paušálně, podle doby (délky použití), podle uskutečněných transakcí atd.

V čem jsou přínosy?

- využívá se "economy of scale"
  - malým uživatelům se nevyplatí kupovat si drahý SW
  - pořídí si jej ASP
    - jeho použití "prodává" více "malým" uživatelům
  - obdobně pro průběžné náklady na správu, …
- pro zákazníka:
  - drahý SW se stává dosažitelný i pro "malé" uživatele
  - zákazník se "neupisuje na dlouhou dobu"
    - když mu služba přestane vyhovovat, přestane ji využívat
      - nenese žádné jednorázové investice
  - náklady zákazníka jsou dobře predikovatelné
    - nejčastěji lineární
  - dostupnost služby může být smluvně zajištěna
    - smlouvami SLA

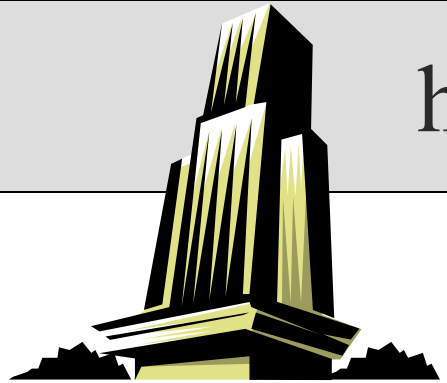
HW jako služba

maximum vlastnictví na uživateli

maximum vlastnictví na poskytovateli

- tradiční přístup k HW:
  - uživatel si pořizuje HW do svého vlastnictví, sám si ho provozuje (u sebe), sám se o něj stará
- alternativa: **server housing**
  - uživatel umístí svůj vlastní server do prostor svého poskytovatele připojení
    - hlavně kvůli lepší konektivitě
    - server stále patří uživateli
    - o server se stará jeho vlastník/uživatel
- alternativa: **server hosting**
  - server patří poskytovateli, je umístěn v jeho prostorách, stará se o něj poskytovatel
    - včetně OS a standardních aplikací, utilit atd.
  - uživatel plní server svými daty
    - Web hosting: vystavuje si tam své WWW stránky
- alternativa: **aplikační hosting**
  - poskytovatel se stará o server
    - který mu také patří
  - uživatel si na serveru provozuje své aplikace
    - tj. aplikace patří uživateli
- alternativa: **ASP**
  - aplikace patří poskytovateli, uživatel pouze používá

hostingové služby



Vznikají nové služby:

- **housing**
  - umístění "celých" zařízení uživatele/zákazníka ve vlastních prostorách
- **hosting**
  - umístění dat a aplikací na zařízeních ve vlastních prostorách

Vznikají specificky vybavené prostory pro "housing" a hosting":

- **telehotely, data centra, telehousy, hostingová centra** …..
- jsou vybaveny vším potřebným
  - konektivitou, zabezpečením, ostrahou, napájením, klimatizací atd.

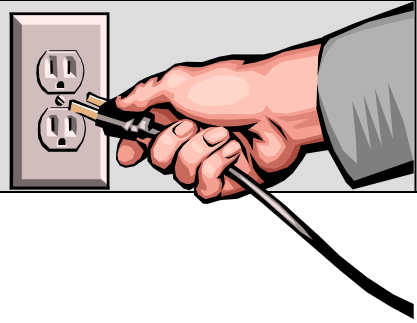
Postupně dochází ke další specializaci i v rámci hostingových služeb:

- **telco operátor**: poskytovatel "datových" služeb (datové okruhy, …)
- **ISP**: poskytovatel (internetové) konektivity
- "**poskytovatel prostoru**" – vlastní prostory, stará se o zabezpečení, napájení, ostrahu, …
- **provozovatel HW** – vlastní HW zařízení (hlavně: servery) a provozuje je
- **provozovatel SW** – vlastní SW vybavení (OS, event. i aplikace) a provozuje je
- ……

mohou různě splývat

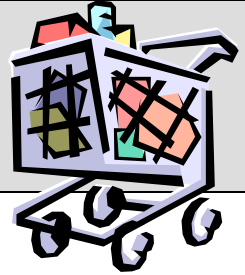
ASP "v čisté podobě"

utility computing



- pozorování:
  - v hostingových centrech (telehousech, …) je dostupné vše (konektivita, výpočetní kapacita, prostor pro data, aplikace, …) v takové míře, v jaké to zákazník požaduje/potřebuje
    - lze průběžně "přidávat" i "ubírat" podle momentální potřeby,
    - bez "pořizovacích nákladů", pouze s lineárními poplatky za objem skutečně využitých zdrojů
- předpoklad:
  - jednotlivé zdroje (výpočetní kapacita, paměť, konektivita, …) jsou tzv. virtualizovány
    - odděleny od své "hmotné podstaty" a nabízeny jako libovolně škálovatelná služba
- důsledek:
  - uživatel může průběžně "konzumovat" zdroje v takové míře, jaká odpovídá jeho momentální potřebám
  - stylem: jako když spotřebovává vodu (elektřinu, plyn, …)
    - pustí si jí tolik, kolik právě potřebuje, platí podle spotřebovaného objemu
- Utility computing:
  - je takový "výpočetní model", kdy zákazník "konzumuje" výpočetní a síťové zdroje na principu "utility" (zdroje typu elektřiny, plynu, vody, …)

on-demand computing



- Výhody virtualizace zdrojů a jejich využití na principu "utility computing":
  - uživatelé (hlavně firmy) nemusí vkládat (větší) kapitálové investice do IT infrastruktury
    - do počítačů, do sítí, do operačních systémů, do "middlewaru"
    - díky ASP ani do aplikací – toho ale využívají spíše menší a střední firmy
  - uživatelé se zbavují rizika neefektivního využití zdrojů
    - toto riziko přenáší na poskytovatele, kteří se s ním dokáží lépe vyrovnat

Princip "utility computing" podporují mnohé velké firmy

- ale často pod jiným názvem:

- IBM: on-demand computing
- HP: adaptive infrastructure
- SUN: N1, computing to n-th degree

- v praxi je zatím zájem o "utility computing" spíše v "interním" provedení
  - velké firmy jej nasazují k efektivnějšímu využití vlastních zdrojů

Parallel Computing

jde více o záležitost architektury počítačů

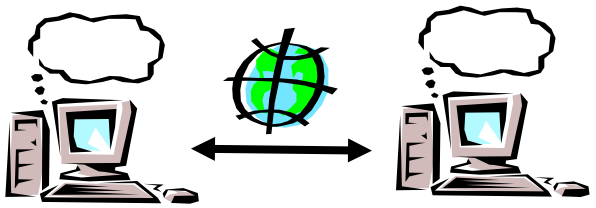
- nikoli počítačových sítí

- zadaný úkol řeší více CPU v rámci jednoho počítače
  - tj. víceprocesorové systémy
- možnost fungování
  - buďto SIMD (Single Instruction Multiple Data), tj. všechny procesory zpracovávají stejným způsobem různá data
    - např. systolické systémy
  - nebo MIMD (Multiple Instruction Multiple Data), tj. každý procesor má samostatný program a zpracovává data různým způsobem
- řešený úkol/problém nemusí mít distribuovanou povahu
  - může být problém s jeho "zparalelněním"
- příklad:
  - grafické algoritmy, rendering
  - signal processing, image processing
  - ….

klasická von-Neumannova architektura počítačů není paralelní, ale sekvenční

Distributed Computing

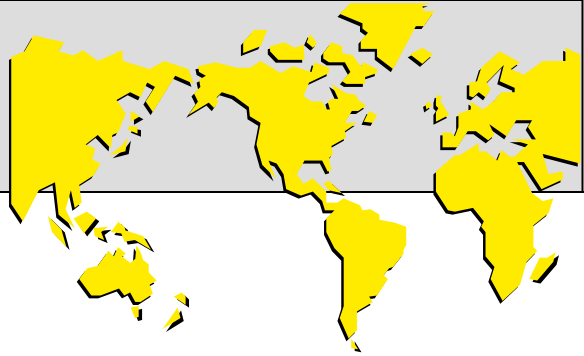
- již se týká sítí
- více samostatných uzlů se vzájemně koordinovaným způsobem podílí na společném řešení zadaného úkolu
  - typicky: spolupracují spolu samostatné (heterogenní) uzly sítě
- řešený úkol/problém má (více) distribuovaný charakter
  - lze jej snadno a přirozeným způsobem rozdělit na stejné či nestejné části
    - charakteru samostatných aplikací, či jejich částí
  - a přidělit samostatným uzlům
- vazba mezi spolupracujícími uzly je volnější
  - než u "parallel computing"



- komunikace mezi spolupracujícími uzly má více asynchronní charakter
- příklad:
  - distribuované databáze
  - transakční a rezervační systémy

distribuovaný může být (bývá) již model klient/server

Grid Computing



- "Grid"
  - znamená mříž, mřížku, rastr, síť, souřadnicovou síť
- Grid Computing je "vyšší stádium" distributed computing
  - výrazně masovější,
  - typicky více homogenní
    - vytváří clustery z menších počítačů

řeší to hlavně problém nedostatečné výpočetní kapacity pro "velké" problémy

- slouží potřebě sdílení výpočetních zdrojů
  - používá se např. pro opravdu náročné úkoly/problémy
- lze si představit jako virtuální superpočítač
  - realizovaný velkým počtem menších zařízení, propojených na malou i velkou vzdálenost
    - vzhledem k dostupným přenosovým rychlostem přestává fyzická vzdálenost prvků Grid-u hrát roli
- příklad:
  - SETI@HOME (využití volné výpočetní kapacity domácích počítačů, pro hledání signálů mimozemských civilizací)

Autonomic Computing

- celkový trend:
  - vše se zvětšuje, stává složitějším a obtížněji řiditelným
    - je problém se správou a managementem "velkých" řešení
- idea: ať mají jednotlivé části větších celků více autonomie
  - ať se dokáží (více) postarat samy o sebe
    - ať jsou vybaveny takovými schopnostmi, které zajistí že budou vyžadovat co nejméně "externích zásahů"
- "self-optimizing"
  - samy optimalizují své fungování, spotřebu zdrojů atd.
- "self-configuration"
  - samy upravují své konfigurační parametry
- "self-healing"
  - samy objevují, diagnostikují a opravují své závady
- "self-protecting"
  - ať se dokáží postarat o vlastní bezpečnost / zabezpečení