

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



Lekce 12: Vývoj výpočetního modelu

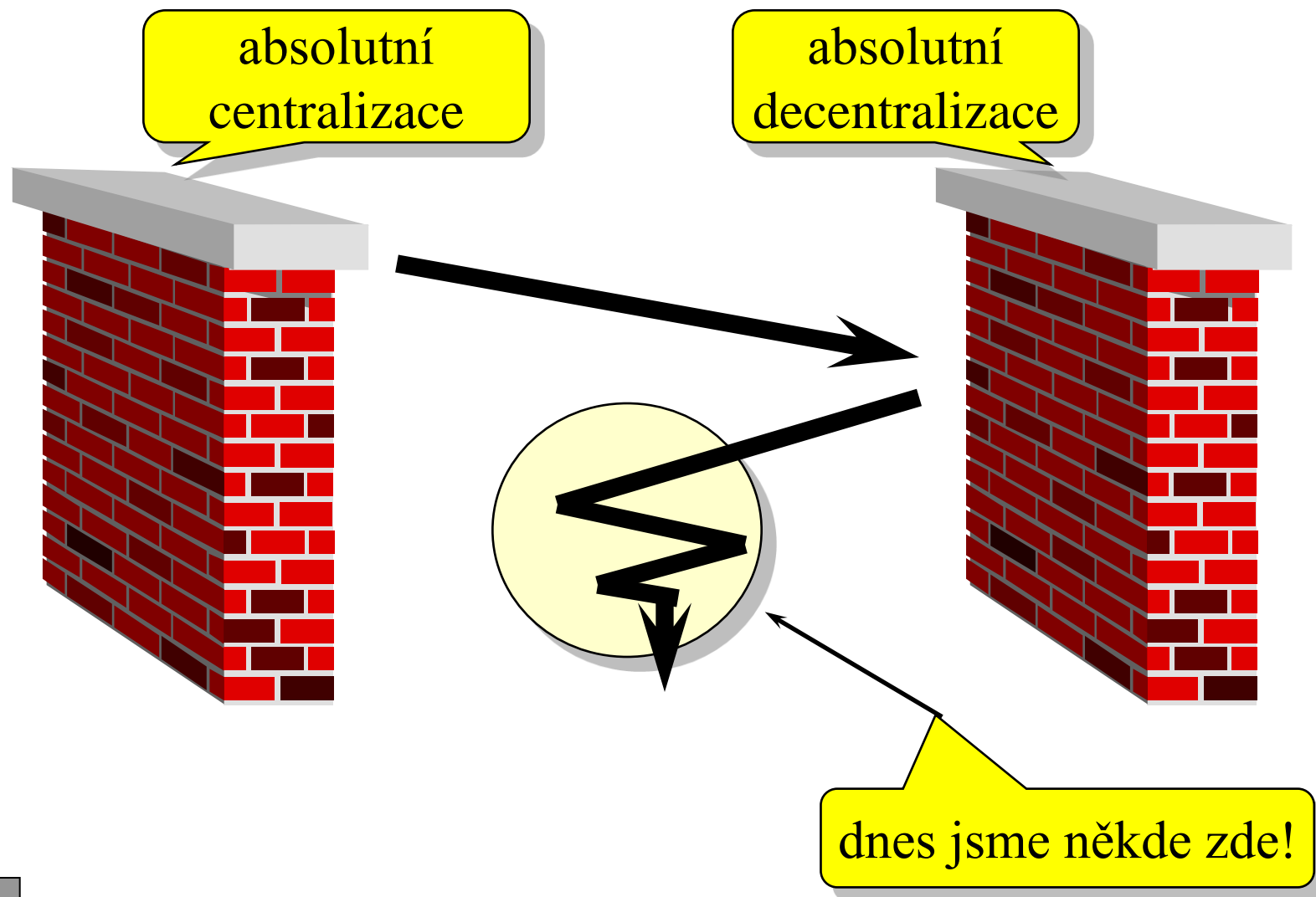
J. Peterka, 2010

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, utility computing, cloud computing ...**)

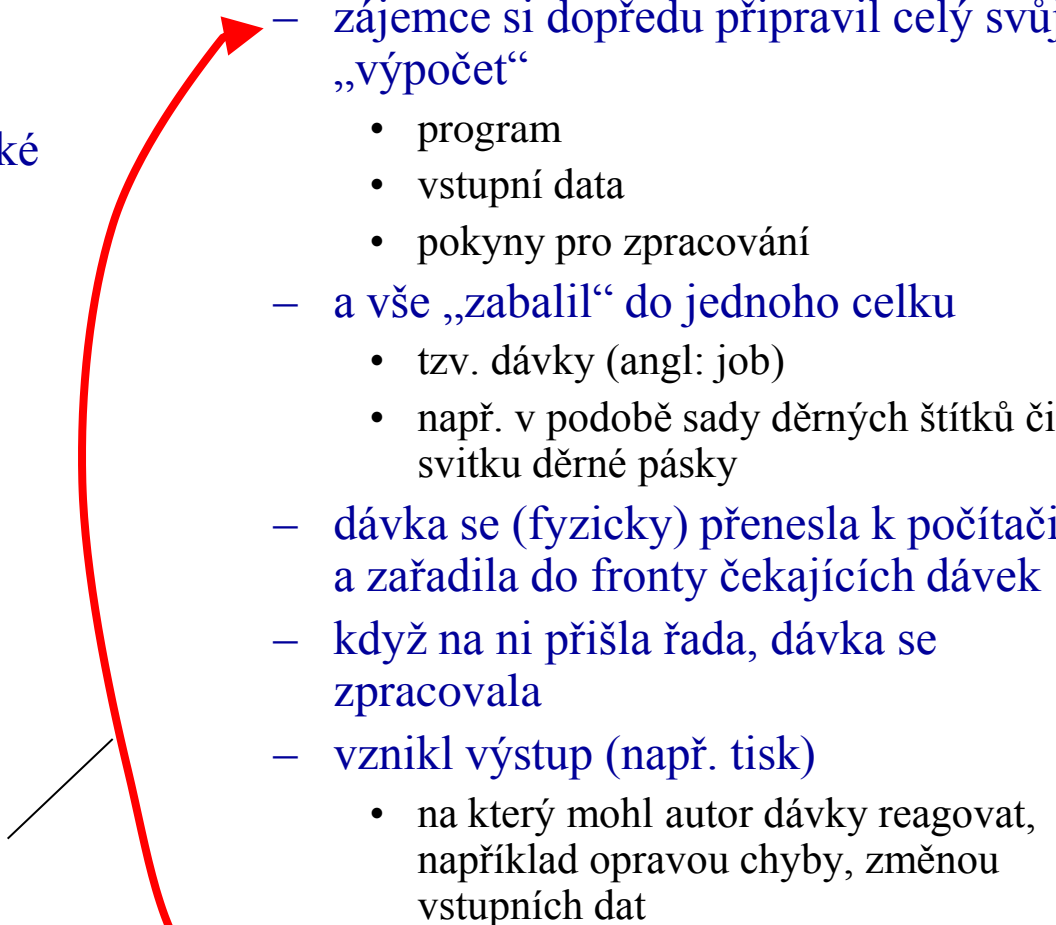
správné pochopení výpočetních modelů je důležité i pro zvládnutí problematiky sítí, pochopení jejich podstaty ...

jak se vyvíjel výpočetní model?

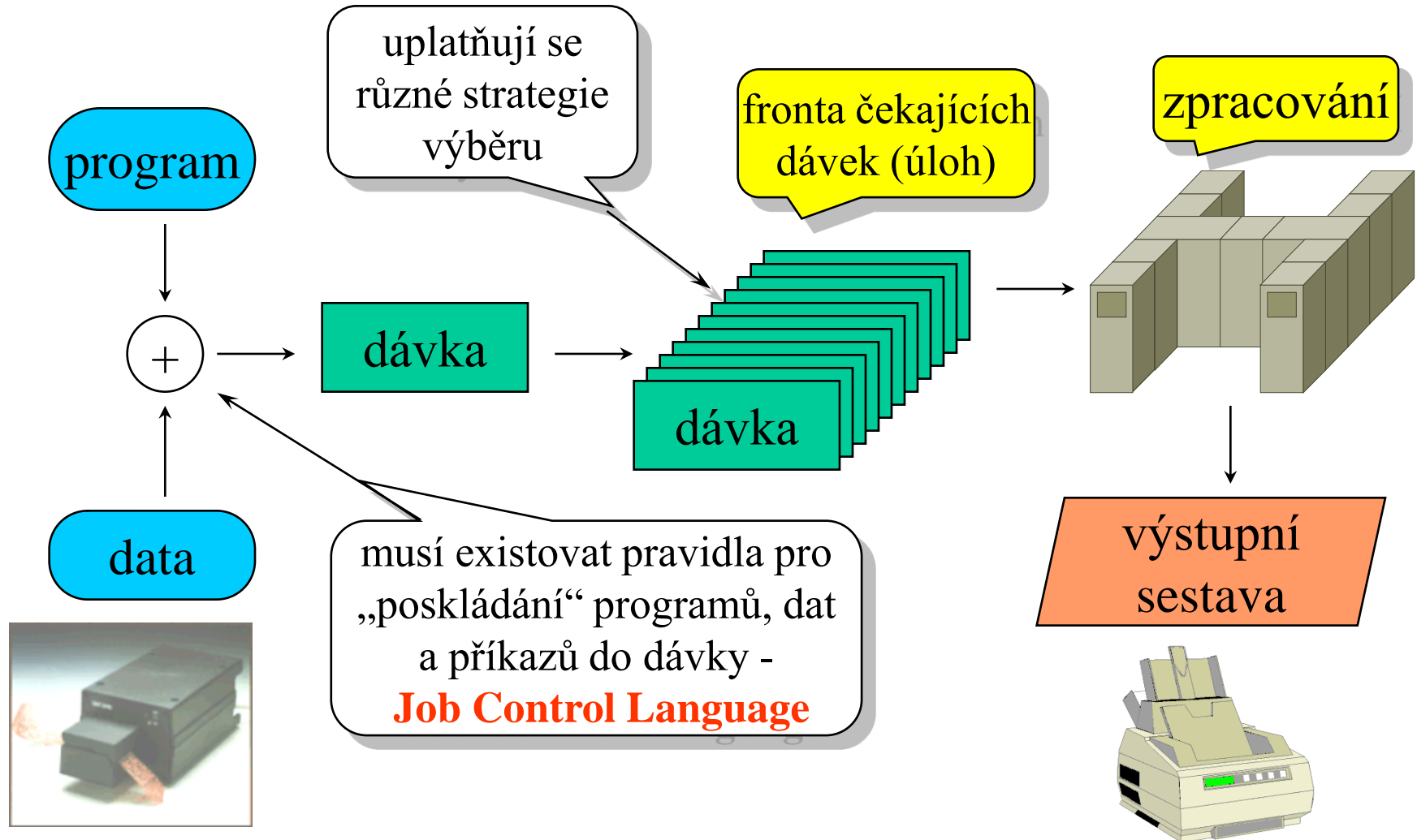


dávkové zpracování (batch processing)

- 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. „obrátko“,
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

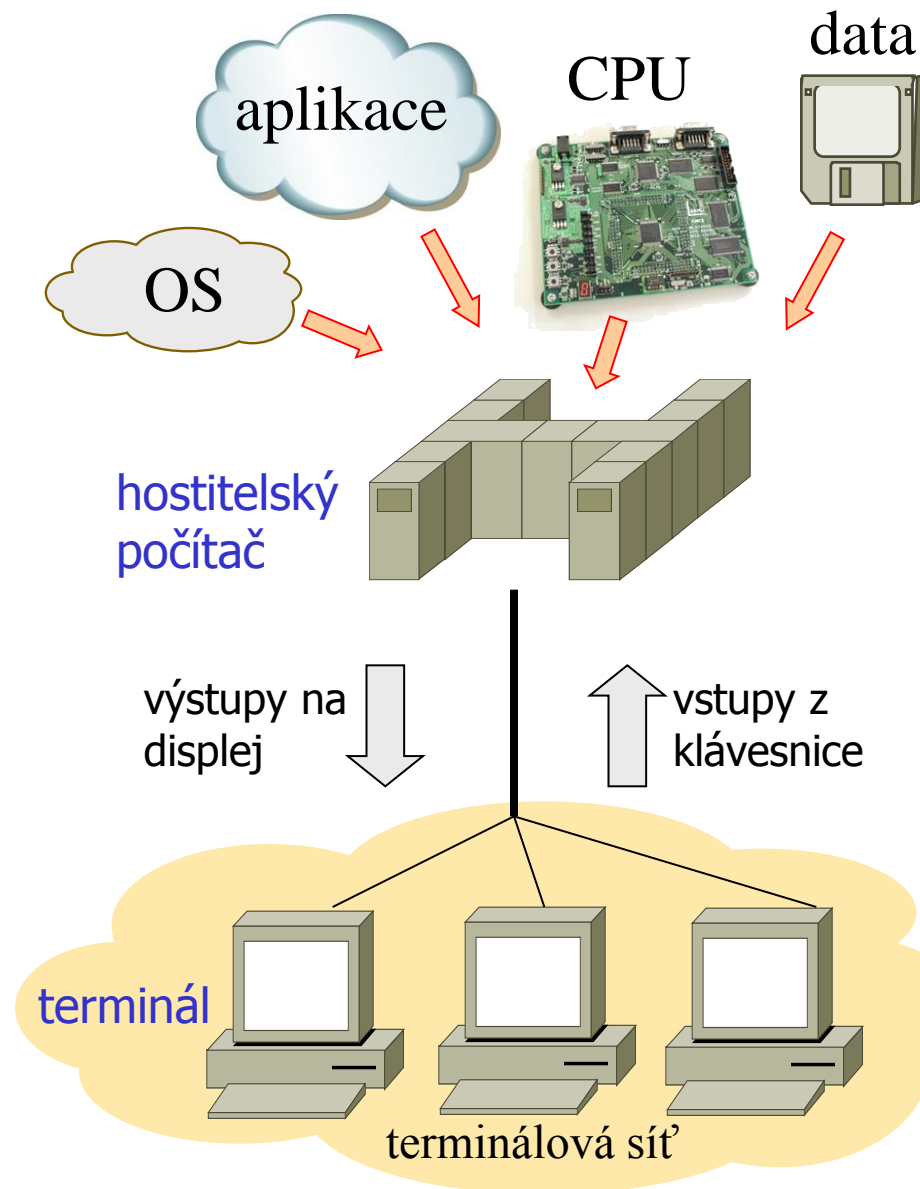
Do budoucna:

- 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 uživatelské 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 uživatelské klávesnice
 - *!!jsou to malé objemy dat, protože se (typicky) pracuje ve znakovém režimu!!*

NEvýhody:

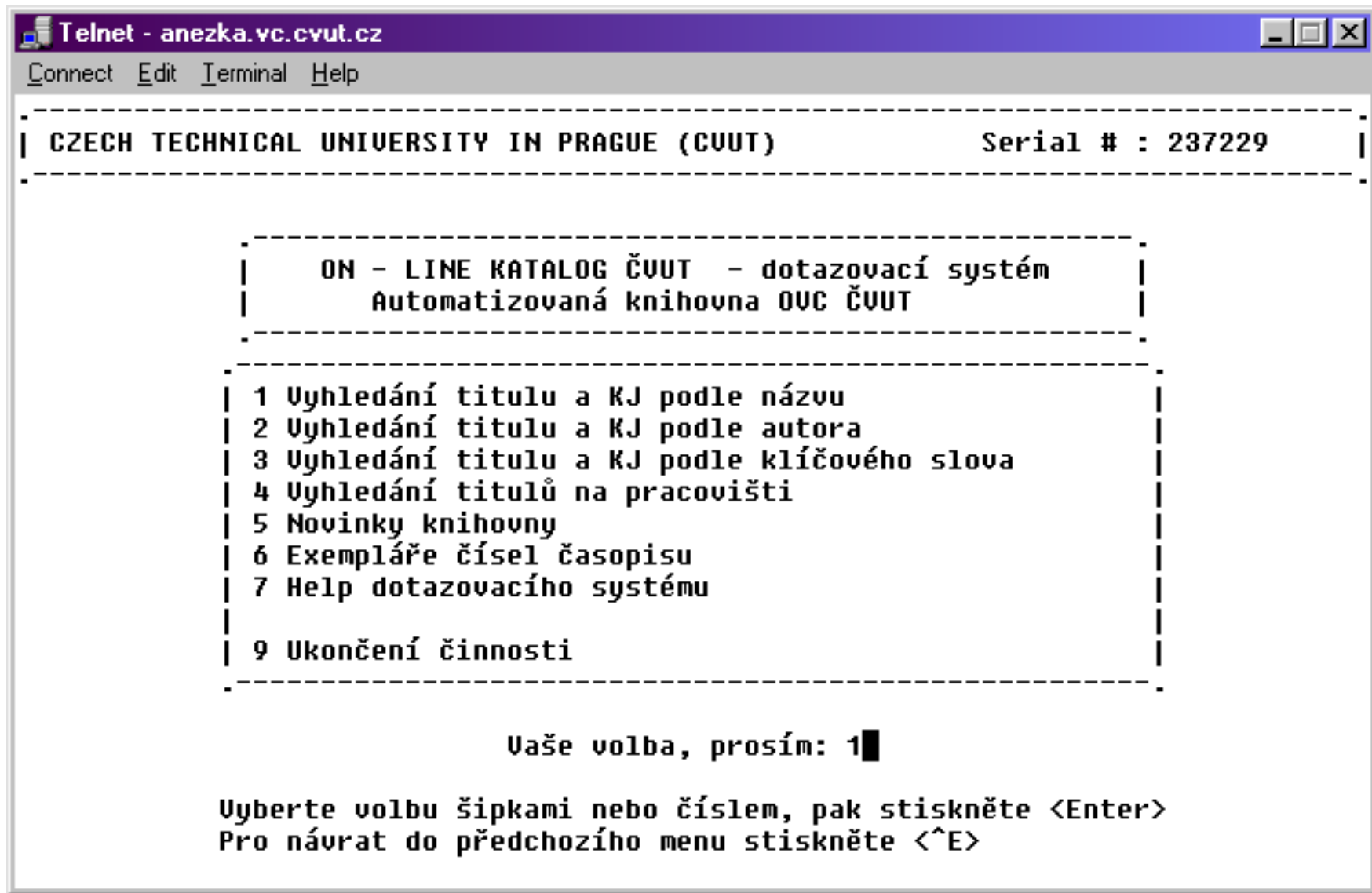
- uživatel má iluzi, že má hostitelský počítač výhradně ke své dispozici
 - ale ve skutečnosti má k dispozici jen n-tou část jeho výkonnosti!
- uživatelský komfort je relativně nízký
 - vzhledem ke znakovému režimu

!!! 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)



```
Telnet - anezka.vc.cvut.cz
Connect Edit Terminal Help
-----
| CZECH TECHNICAL UNIVERSITY IN PRAGUE (CVUT)          Serial # : 237229 |
-----

      -----
      |          ON - LINE KATALOG ČVUT - dotazovací systém          |
      |      Automatizovaná knihovna OVC ČVUT                        |
      -----

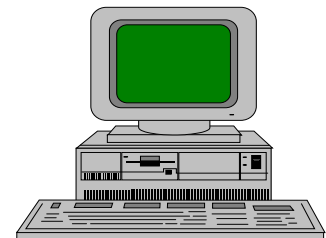
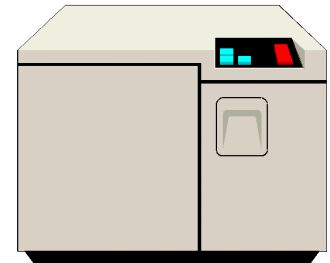
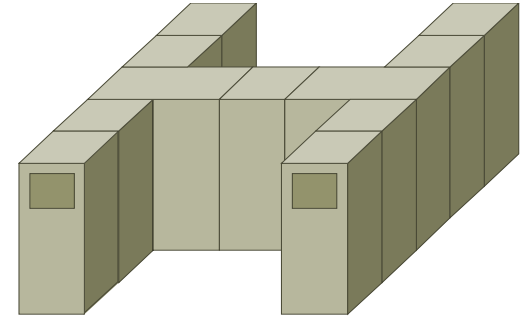
      -----
      | 1 Vyhledání titulu a KJ podle názvu                          |
      | 2 Vyhledání titulu a KJ podle autora                         |
      | 3 Vyhledání titulu a KJ podle klíčového slova                |
      | 4 Vyhledání titulů na pracovišti                             |
      | 5 Novinky knihovny                                           |
      | 6 Exempláře čísel časopisu                                    |
      | 7 Help dotazovacího systému                                  |
      |                                                                |
      | 9 Ukončení činnosti                                           |
      -----

      Vaše volba, prosím: 1

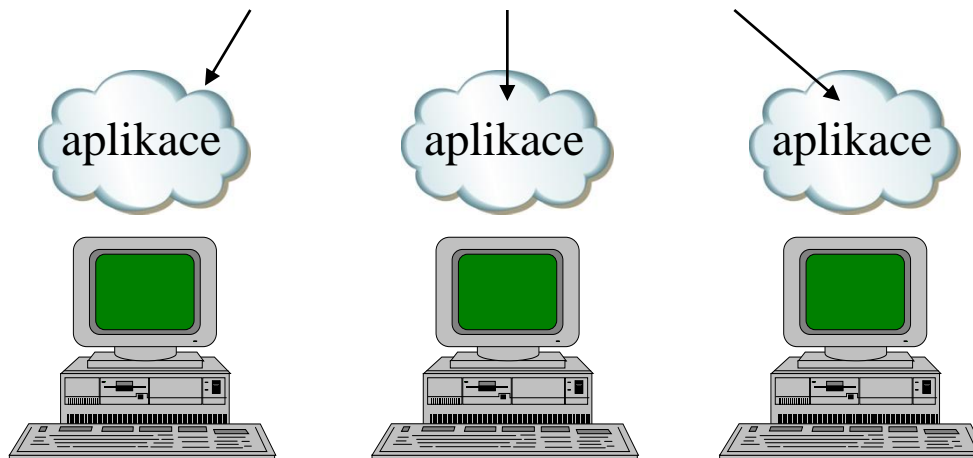
      Vyberte volbu šipkami nebo číslem, pak stiskněte <Enter>
      Pro návrat do předchozího menu stiskněte <^E>
```

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

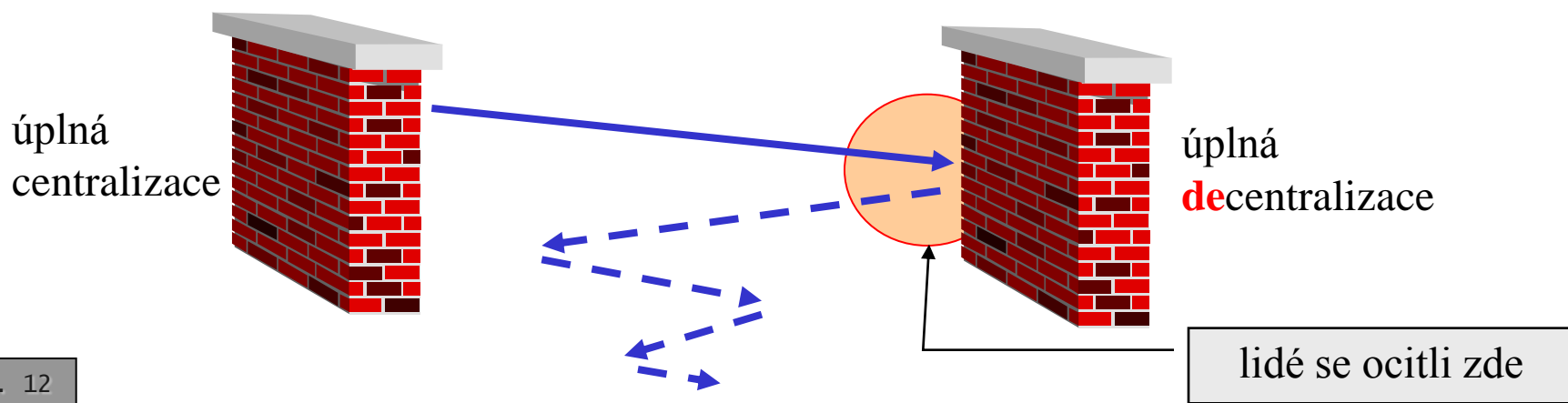


žádná vzájemná vazba



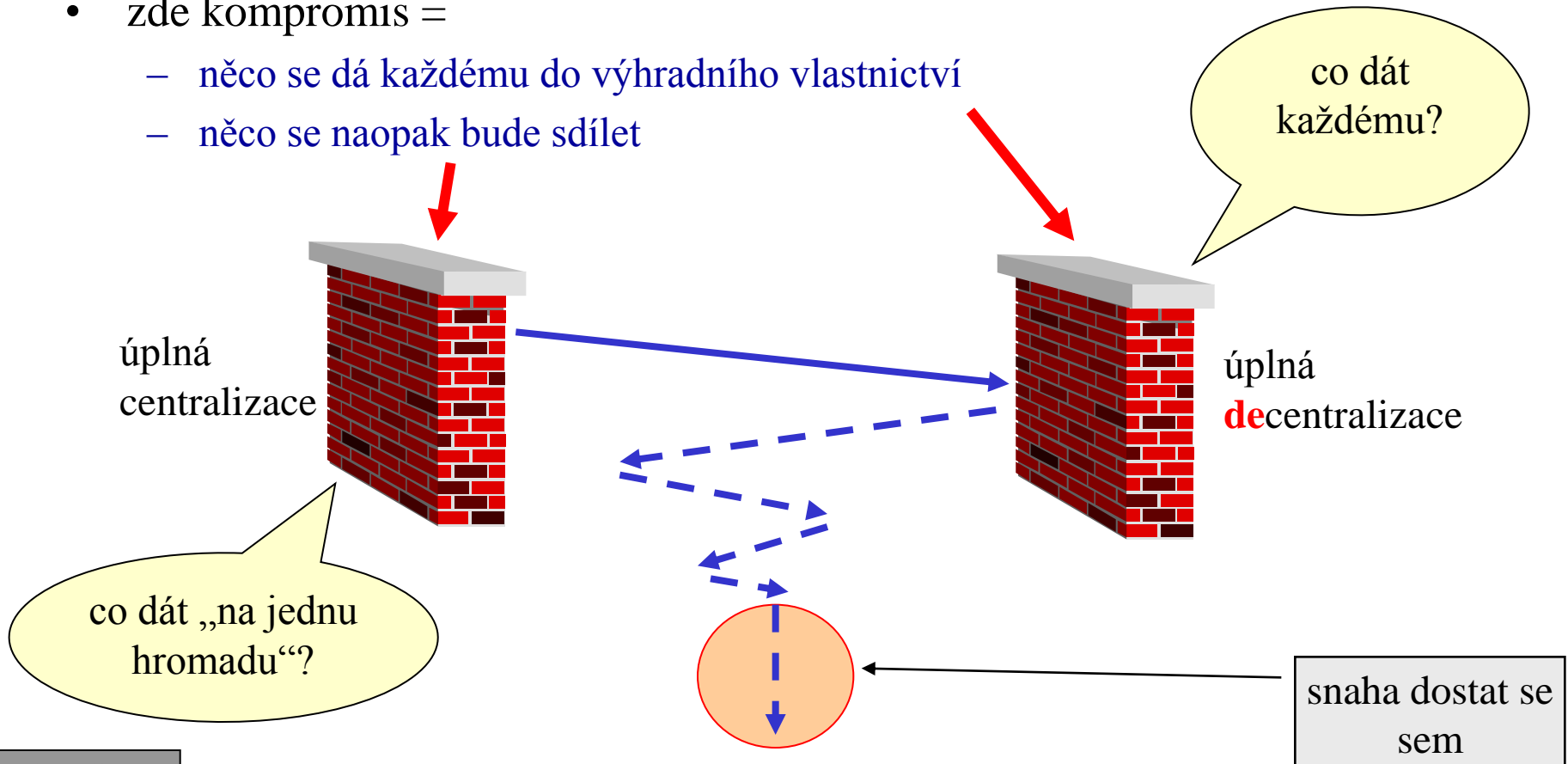
éra izolovaných počítačů

- 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



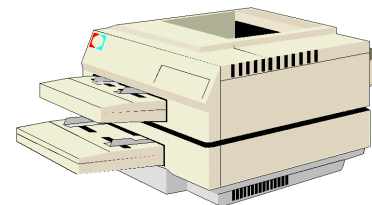
co má smysl ?

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

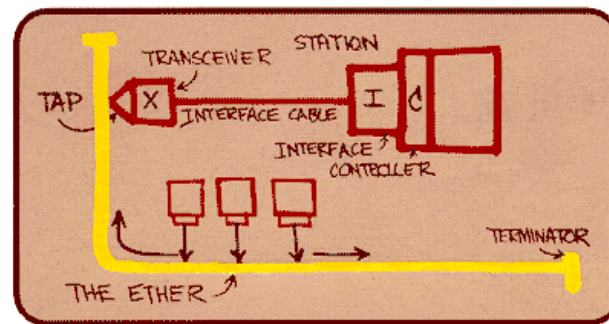
sdílet:

- 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

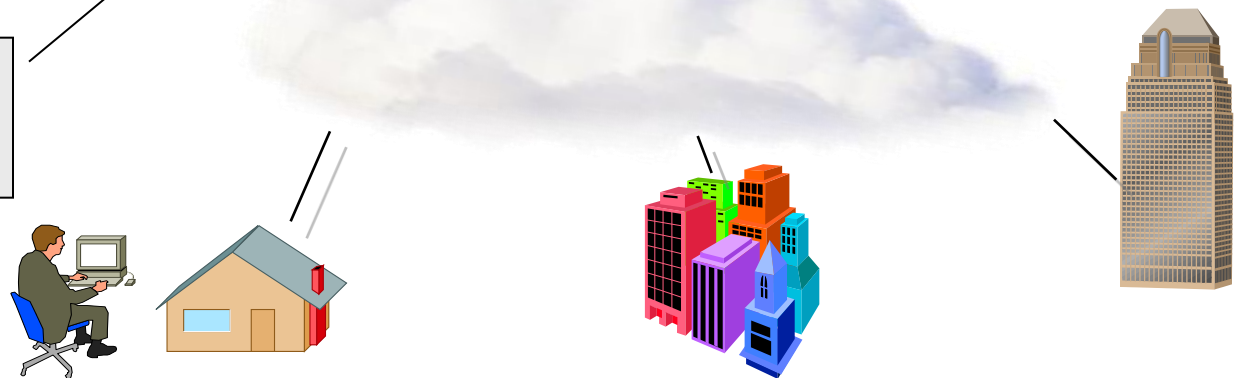
- řeší především potřebu sdílení
 - souborů (programů, dat)
 - periferií (tiskáren,)
- uživatel nesmí sdílení poznat
 - uživatel nesmí pozorovat významnější rozdíl v rychlostech přístupu ke sdíleným a privátním objektům
 - je vhodné, když si uživatel vůbec nemusí uvědomovat fakt sdílení
 - mechanismy 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“

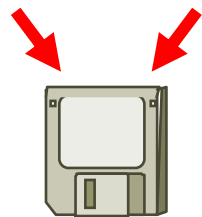
přestává platit až se
zaváděním broadbandu



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

data + aplikace

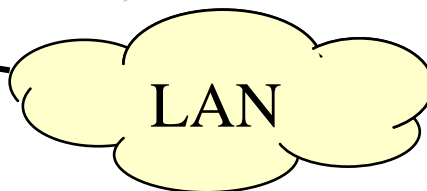


umístění,
jako soubory

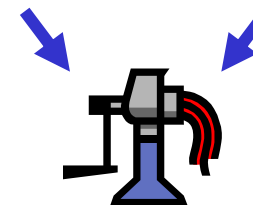


file server

důsledek: celé
aplikace a
všechna data se
musí přenášet



data + aplikace



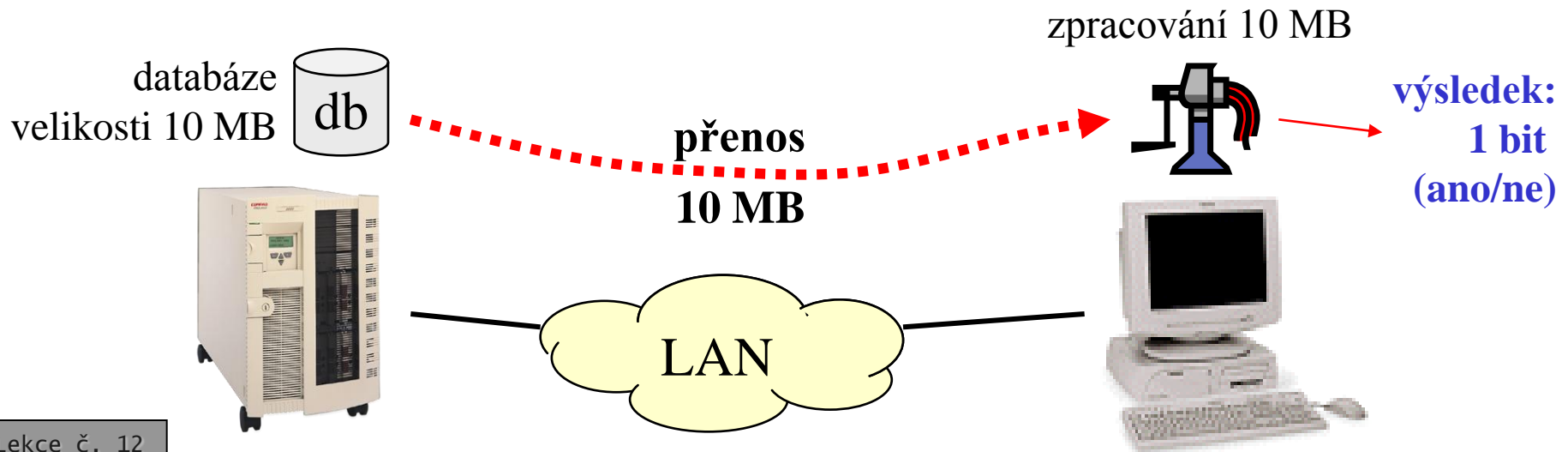
běh,
zpracování



pracovní stanice

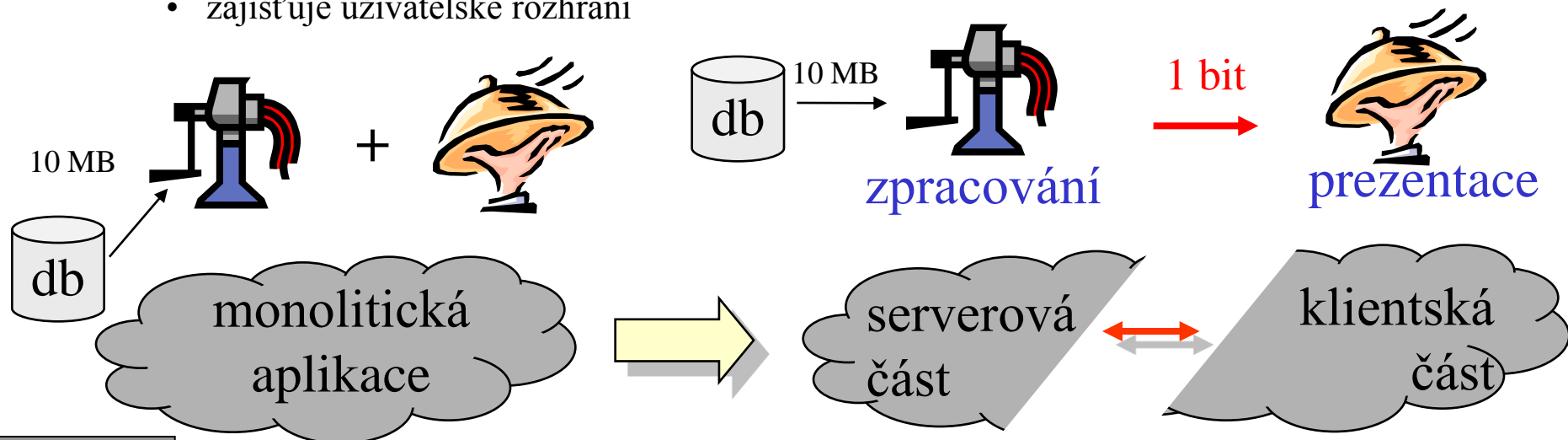
model file server/pracovní stanice

- 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 zpracovávána 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 sítí
- klient a server mohou stát na různých platformách



představa modelu klient/server



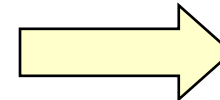
- komunikace mezi klientem a serverem se odehrává stylem: požadavek/odpověď
 - server pasivně čeká, až dostane nějaký požadavek.
 - komunikaci iniciuje klient, zasláním požadavku
 - musí být definována vzájemná komunikace mezi klientem a serverem
 - komunikační protokol (např. HTTP)
- mnoho služeb dnes 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**

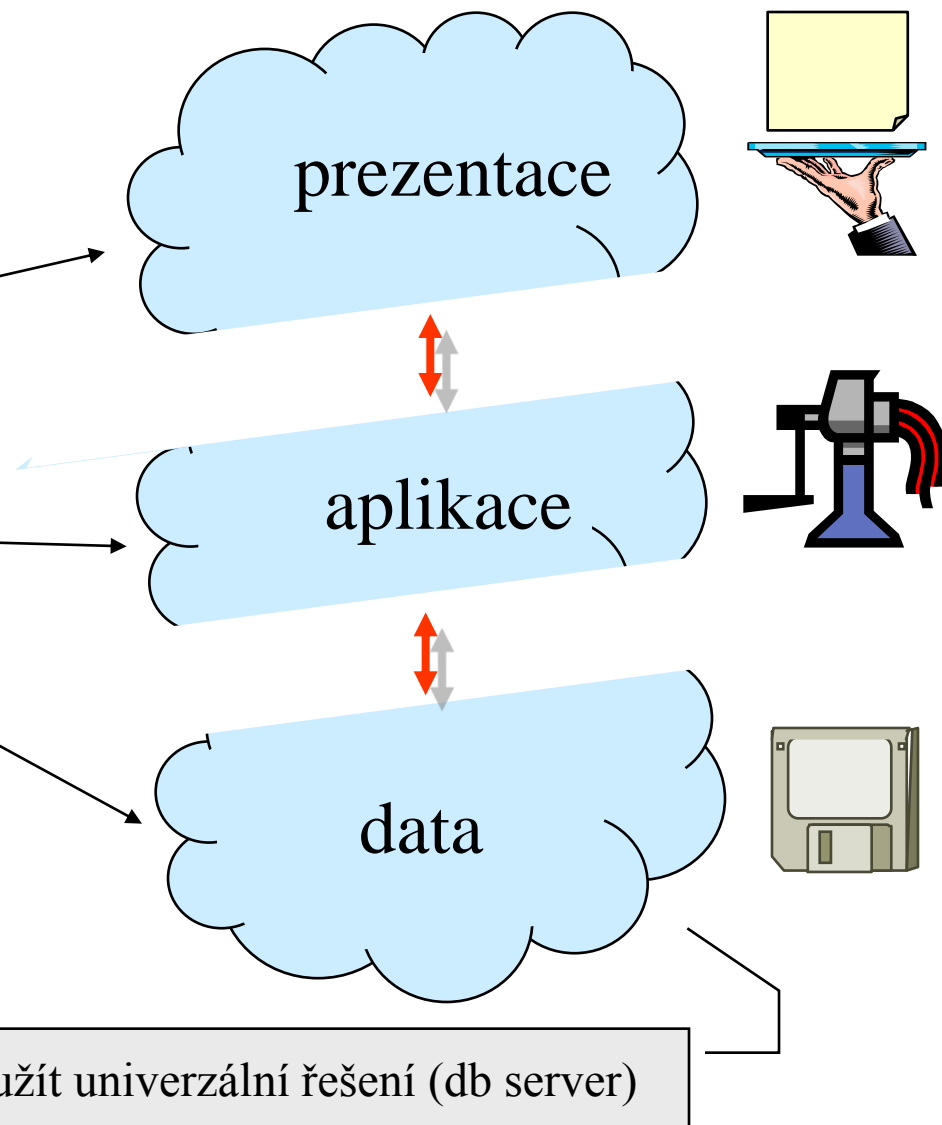
- vlastní logika aplikace

- **správa dat**

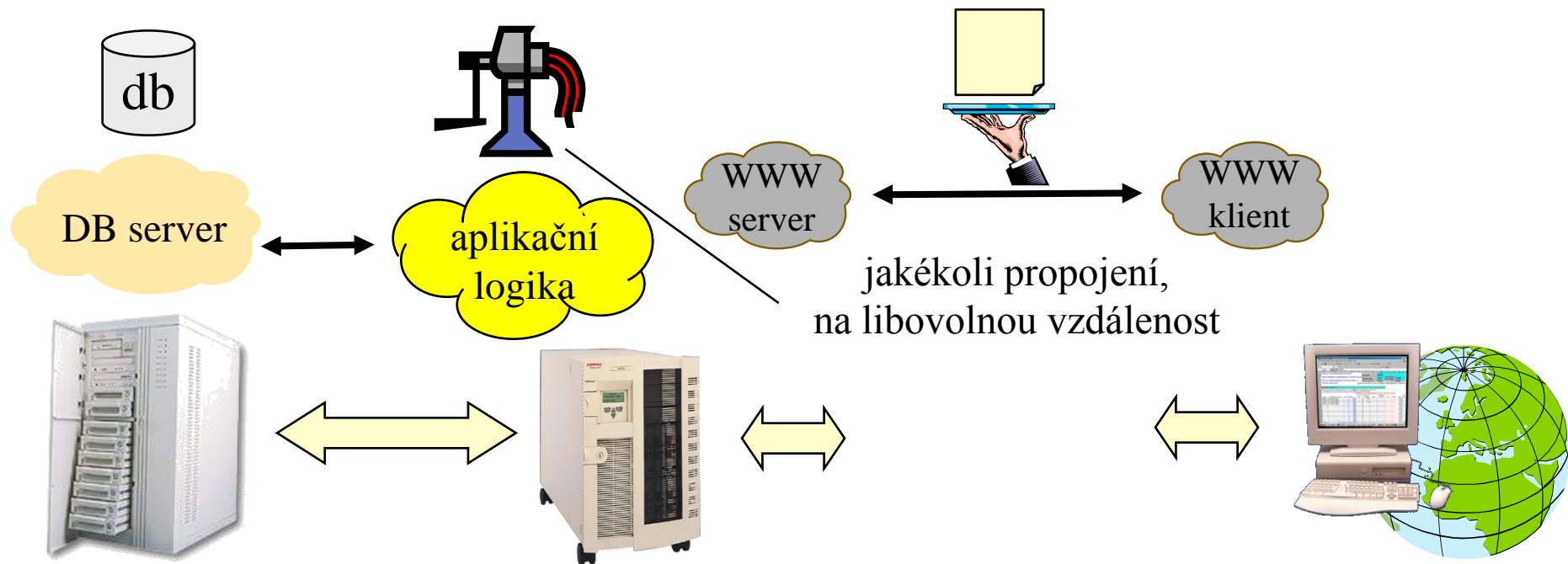
- vlastní databázové operace

- lze implementovat jako:

- 3 úrovňové řešení
- 2 úrovňové řešení (celkem 5 možností)

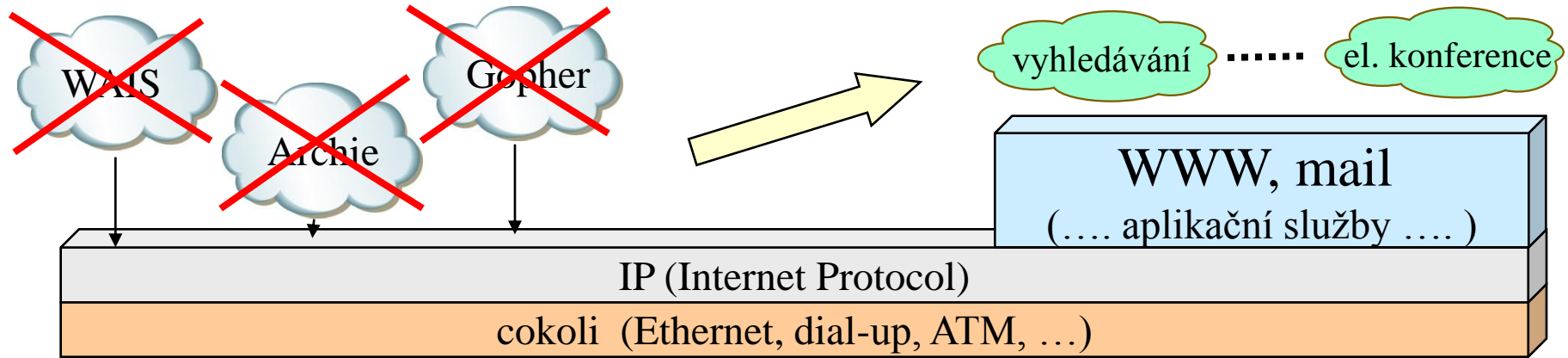


představa 3-úrovňové klient/server aplikace



- 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

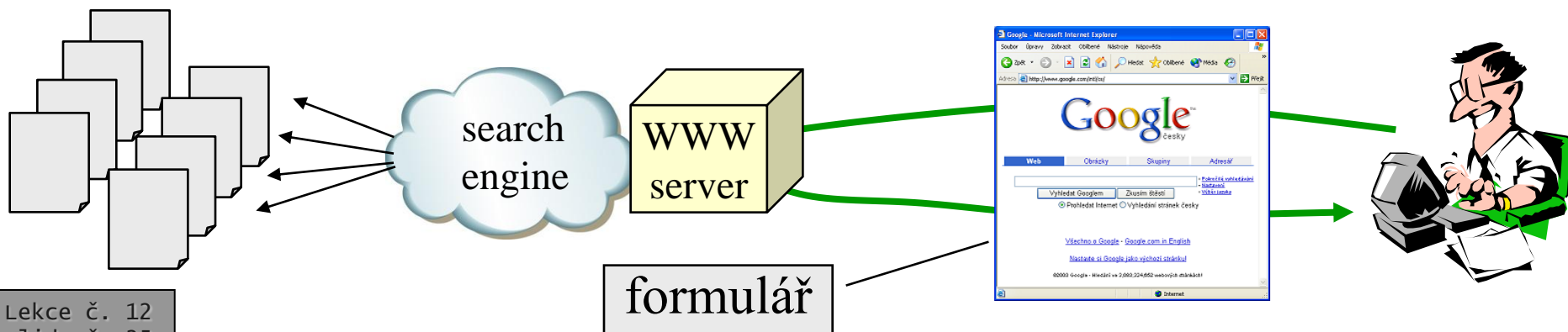
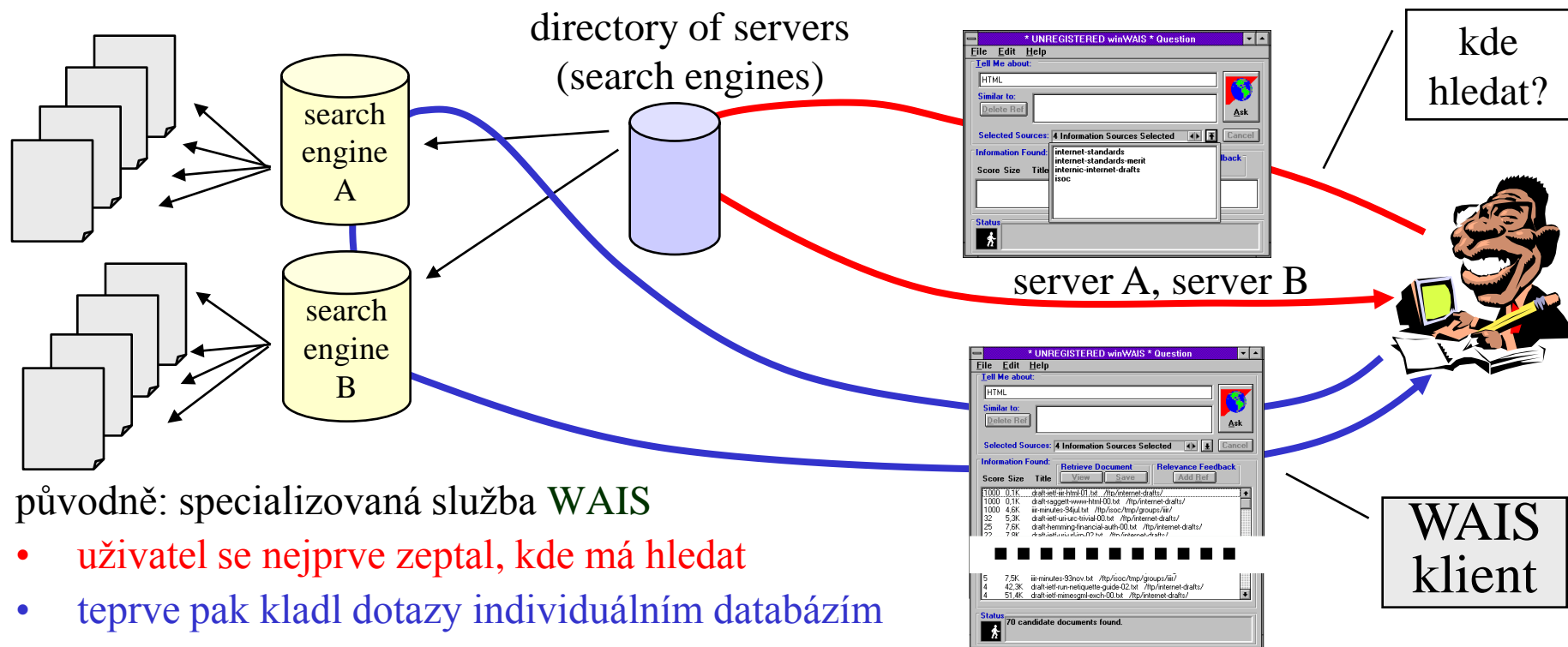
důsledky



- 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, Čmucha atd., dnes Google, AltaVista, Jyxo ...
 - informační (a další) on-line služby
 - např. Obchodní rejstřík, přímé bankovníctví (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



"Tlusté PC" vs. tenký klient

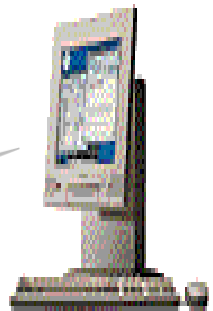
směr dalšího vývoje:

- 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"

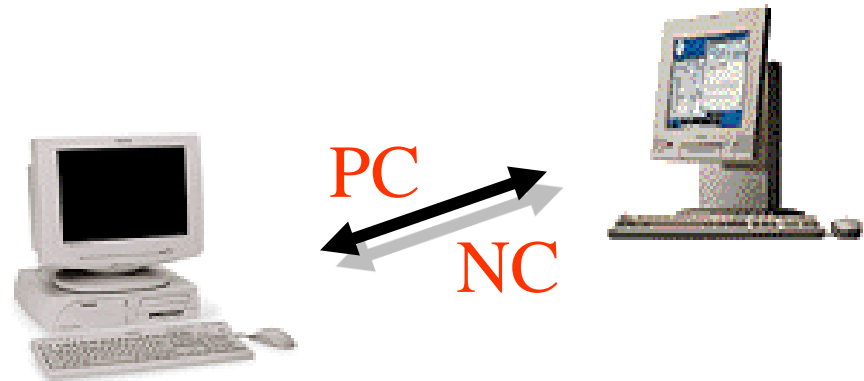


tenký klient
(thin client)

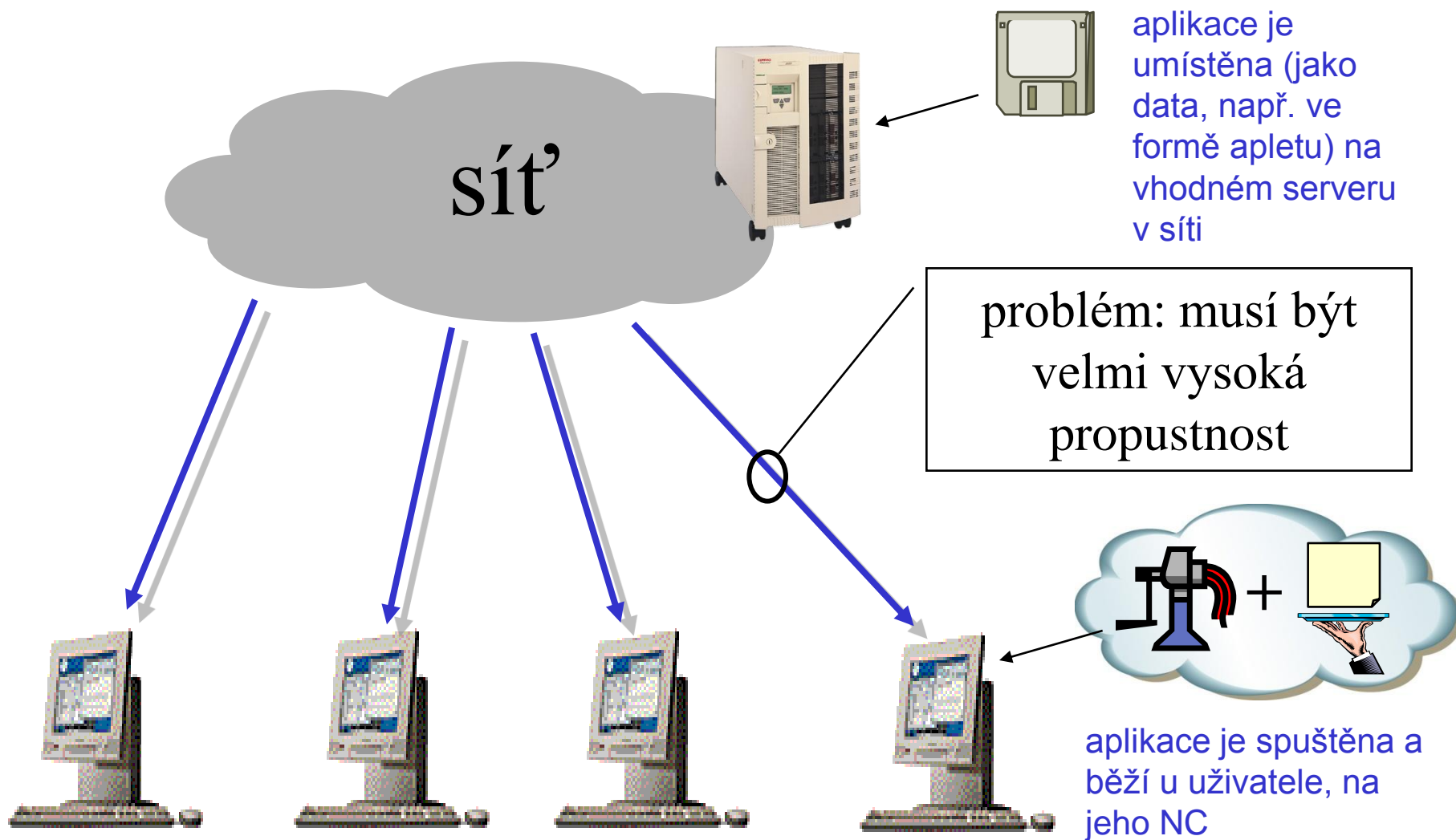


jak realizovat tenkého klienta?

- 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. applety 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

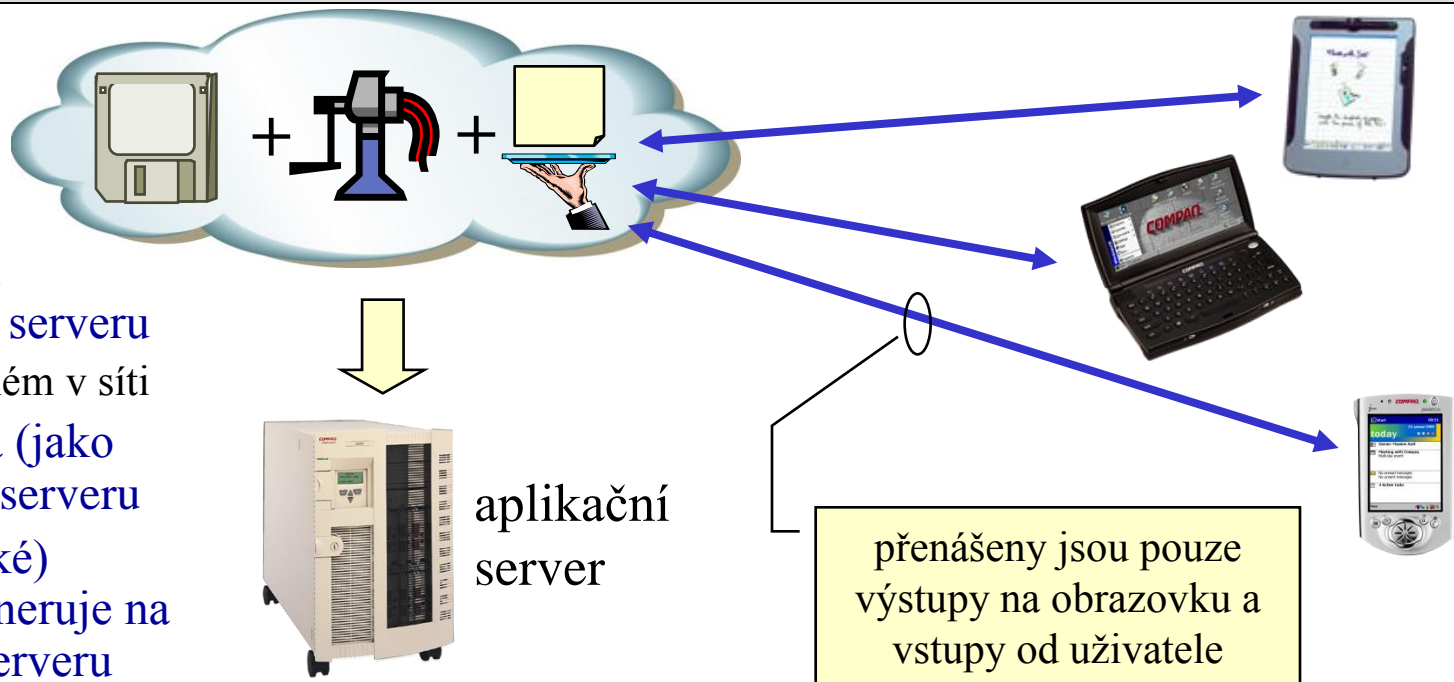
- cesta snižování TCO (nákladů na provoz) skrze NC se ukázala jako nepříliš schůdná
- další pokus se ubíral cestou návratu k plné centralizaci
 - návratu k modelu host/terminál
 - ale bez jeho problémů s nízkou uživatelskou přítulností
- další motivace:
 - snaha umožnit použití i jiných zařízení než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

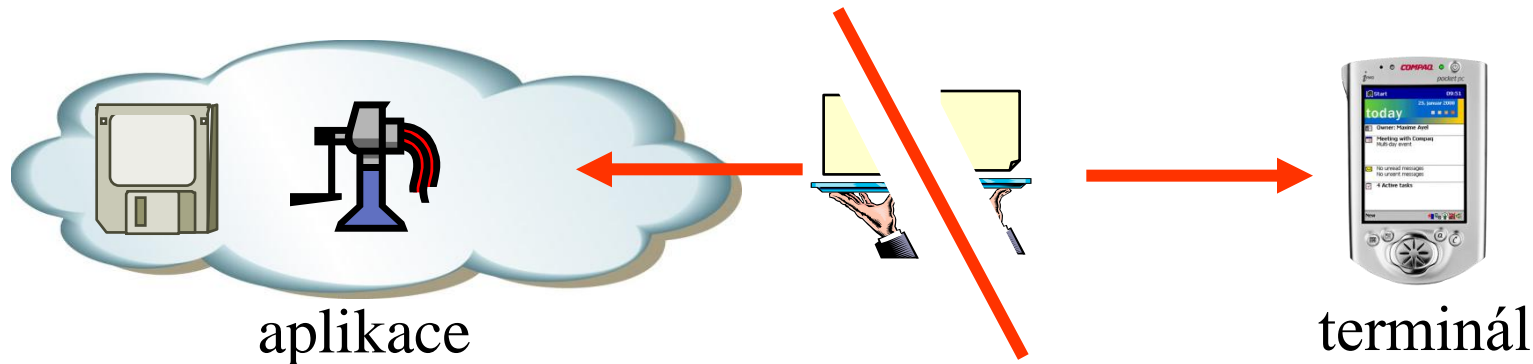
- aplikace:

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



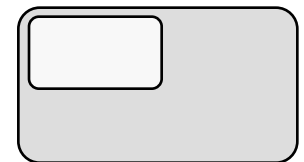
- v principu se jedná o návrat k původnímu modelu host/terminál
 - snahou je využít všech výhod centralizace ke snížení nákladů na provoz a správu (TCO)
 - ale bez ztráty komfortu pro uživatele (nutnost fungování v grafickém režimu)
- problém je v tom, že generovaná grafická data mohou být neúnosně velká,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

Server-Based Computing představa realizace

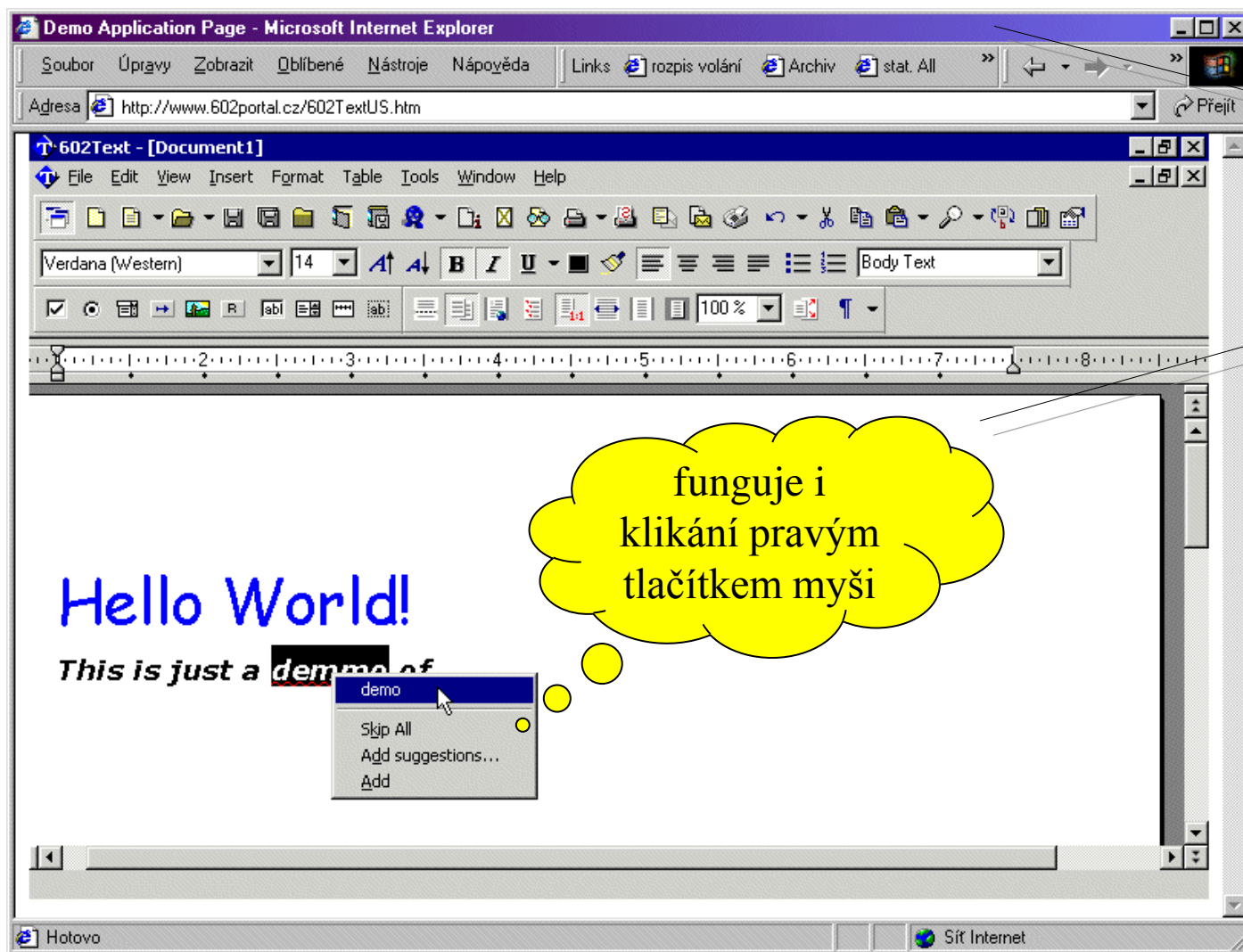


- řešením je vhodné „roztržení“ prezentačních funkcí
 - grafického subsystému („toho, co generuje grafická data“)
- a přemístění části generující grafická data přímo do terminálu
 - tak aby se objemná grafika generovala „místně“, a nemusela se nikam přenášet
 - lze se lépe přizpůsobit místním možnostem zobrazení
- „řez“ se musí udělat s ohledem na:
 - minimalizaci objemu přenášovaných 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)



WWW
browser

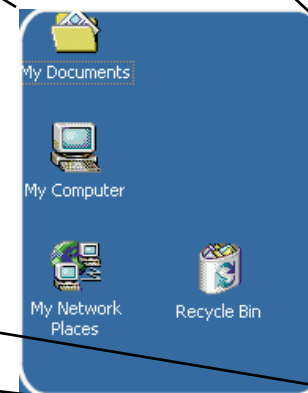
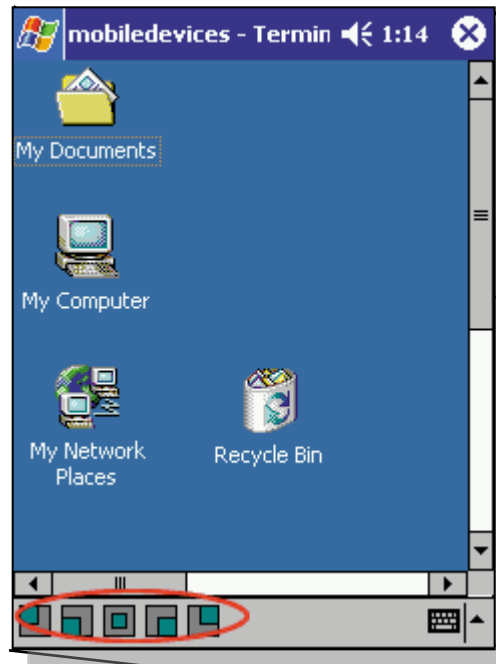
aplikace
(textový
editor),
běžící na
vzdáleném
počítači

funguje i
klikání pravým
tlačítkem myši

příklad: Terminal Services na PDA

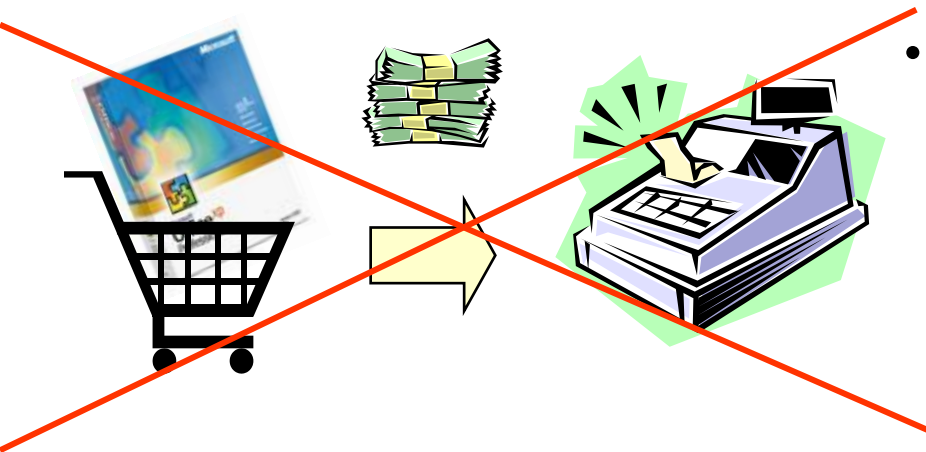


- disproporce mezi velikostí "virtuální pracovní plochy" a velikostí reálného displeje se řeší skrze tzv. panning
 - reálný display ukazuje jen výřez virtuální pracovní plochy



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

SW jako služba



- princip **ASP** (Application Service Providing)
 - uživatel si SW nepožizuje 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 !!!

- 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!

- 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

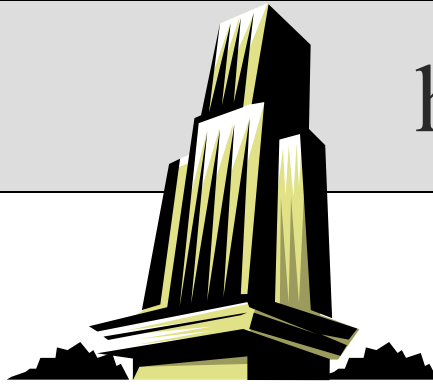
maximum
vlastnictví
na uživateli



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 prostorech, 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.

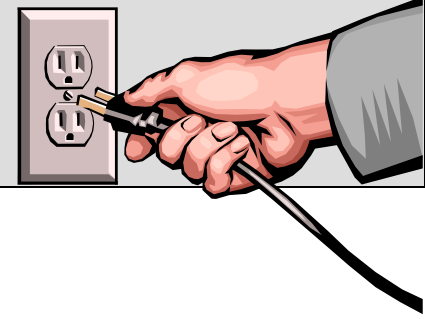
Postupně dochází ke další specializaci i v rámci hostingových služeb:

- **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 (elektrinu, 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 elektriny, plynu, vody, ...)

příklad: virtuální PC

cloud computing



- je to nový „buzzword“, který nemá zcela přesné vymezení, a který integruje (zastřešuje) mnoho z předchozích konceptů a výpočetních modelů, včetně:
 - utility computing:
 - „konzumuji služby tehdy, kdy chci, a v takovém objemu, v jakém potřebuji“
 - » jako elektřinu, vodu, plyn
 - libovolně škálovatelné
 - » ale za úplatu
 - zahrnuje virtualizaci:
 - mohu využívat HW zdroje nezávisle na jejich „hmotné podstatě“
 - » paměť, výpočetní kapacitu, přenosovou kapacitu
 - zahrnuje hostingové služby (ASP,)
 - někdo vlastní a udržuje HW a SW, já je pouze využívám
 - server based computing, případně i network centric computing
 - nemusím mít aplikace u sebe (a vlastnit je)
 - » běží přímo v „cloudu“ (server based computing), případně si je stáhnu a spustím u sebe (Network Centric), nebo

cloud computing - příklady

pro jednotlivce:

- webmaily (například Gmail, Hotmail, Yahoo Mail apod.)
 - emailový klient, uživatel používá „na dálku“ (přes web), nestará se o instalaci, upgrady, umístění své schránky atd.
- síťová úložiště (Humyo, ZumoDrive, SkyDrive, S3 atd.)
 - chovají se a používají obdobně jako místní disk, ale jsou umístěny „někde v síti“
- on-line kancelářské balíky (Google Docs, MS Office Live, ...)
 - uživatel používá běžné kancelářské aplikace „na dálku“, nemusí instalovat ani vlastnit aplikaci

pro firmy:

- celá „prostředí“ pro provozování aplikací, poskytování služeb, vyžívání zdrojů
- obvyklé členění:
 - **SaaS** (Software as a Service)
 - obdoba ASP – software jako služba
 - **PaaS** (Platform as a Service)
 - poskytování virtualizovaného výpočetního prostředí (platformy) formou služby
 - virtualizované servery pro provozování aplikací
 - **IaaS** (Infrastructure as a Service)
 - poskytování virtualizovaných síťových zdrojů formou služby
 - konektivita a paměť podle momentální potřeby
- například:
 - Windows Azure (Microsoft)
 - Web Services (Amazon)
 - App Engine (Google)
 - Force.com

cloud computing - shrnutí

- základní principy cloud computingu:
 - jde o kombinaci výpočetních modelů, ale také ekonomických řešení
 - zdroje vlastní poskytovatel služby, uživatel platí za používání
 - zdroje jsou „v oblaku“ (in the cloud)
 - HW zdroje, SW zdroje, data
 - uživatel se nestará kde konkrétně (nepotřebuje to vědět)
 - přístup k těmto zdrojům je obecně odkudkoli
 - zdrojů je tolik, kolik je potřeba (plná škálovatelnost – jen otázka peněz)
 - uživatel pouze používá
 - uživatel se nestará o pořízení, instalaci, zprovoznění, upgrady, patche atd.
 - uživatel se nestará o zálohování, zabezpečení,
 - zdroje se používají (většinou) na dálku
 - prostřednictvím webu, případně jinak (vzdálený přístup)
 - možné je i stažení aplikace (appletu, „kusu kódu“) a její běh na koncovém zařízení
 - využívá se „economy of scale“
 - jeden poskytovatel má typicky více zákazníků současně

