

# Lekce 10: Vývoj výpočetního modelu

*Jiří Peterka*

# co je výpočetní model?

- ucelená představa o tom,

- kde jsou aplikace uchovávány
    - jako programy (spustitelné soubory)
  - kde se nachází / kde vznikají data
    - jako (datové) soubory, streamy, ...
  - zda jsou aplikace rozděleny na části (a jaké)
    - kde se tyto části nachází, jak vzájemně spolupracují
  - kde se zpracovávají a uchovávají data
    - nemusí to být tam, kde data vznikají či kam jsou ukládána
  - kde se nachází a co dělá uživatel
    - zda je/může být v interaktivním kontaktu se svou aplikací
    - zda s aplikací pracuje „na dálku“
    - ....

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

- výpočetní model je závislý

- na možnostech HW a SW
  - na (ne)dostupnosti sítí a propojení
  - na preferencích uživatelů i výrobců
  - na snahách minimalizovat náklady
  - na potřebách bezpečnosti
  - ....

- výpočetní model se stále vyvíjí

- některé výpočetní modely nepočítají s existencí sítě
    - např. dávkové zpracování
  - jiné výpočetní modely spíše počítají s existencí sítě
    - např. model klient/server
  - další modely již vyžadují existenci sítě
    - např. distribuované zpracování, network-centric computing, thin-client, server-centric computing, utility computing, cloud computing ...

# dávkové zpracování

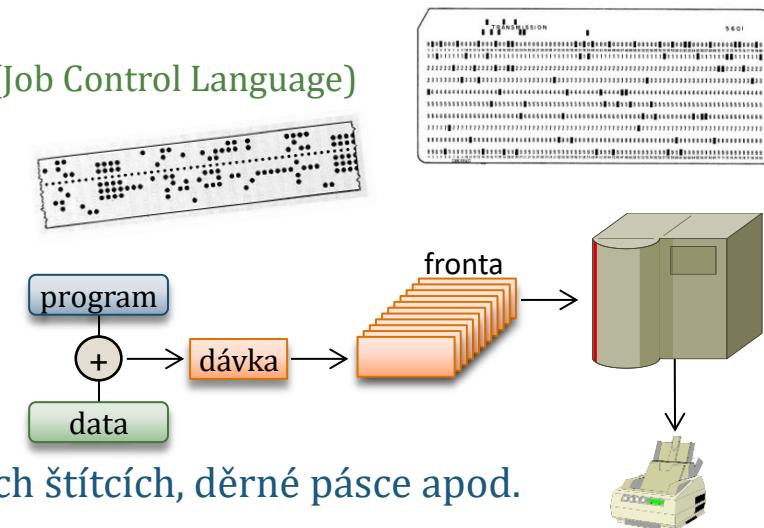
- **nejstarší výpočetní model**

- HW byl drahý a pomalý, zájemců o využití bylo hodně
  - vše muselo být nějak sdíleno
- ale nebylo možné, aby:
  - uživatelé pracovali s počítačem současně
    - nebyl na to ještě vhodný SW, který by umožnil nějaké sdílení, ani HW nebyl dost výkonný
  - uživatelé měli přímý (interaktivní) kontakt se svou aplikací
    - nebyly ještě k dispozici terminály (pracoviště, s klávesnicí a tiskárna/displejem)



- **podstata dávkového zpracování (batch processing)**

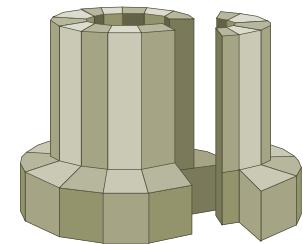
- uživatel musel dopředu „říci, co všechno chce“
  - musel připravit programy, data i pokyny pro jejich zpracování, a „zabalit“ je do jednoho celku
    - tzv. **dávky**, anglicky: batch, job
      - dávka je sestavena („zabalena“) pomocí jazyka JCL (Job Control Language)
      - nejčastěji: ve formě děrných štítků či děrné pásky
  - dávky od různých uživatelů se řadily do front
    - s různými režimy či strategiemi
      - s cílem max. vytížit dostupné zdroje
  - dávka byla zpracována tehdy, když na ni došla řada
    - aniž by uživatel/autor dávky „byl při tom“
  - výstupem byla tisková sestava, sestava na děrných štítcích, děrné pásky apod.



# RJE a model autonomních agentů

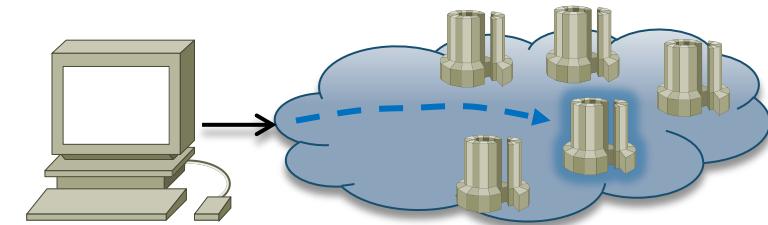
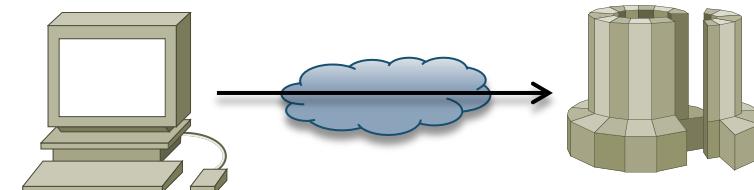
- **výhoda dávkového zpracování:**

- dokáže velmi efektivně využít „to, co je k dispozici ....“ (hlavně HW)
  - proto dodnes není tento model úplně mrtvý, ale někde se nadále používá
    - např. u superpočítačů, pro výpočetně náročné úlohy



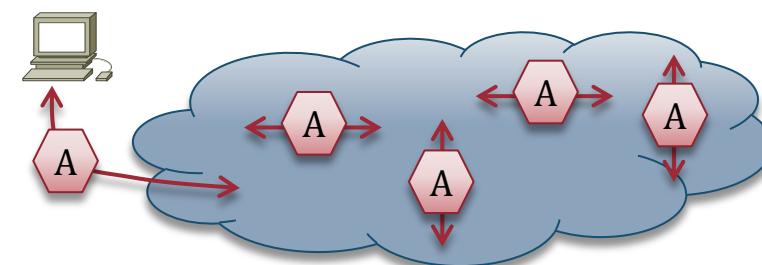
- **novější podoba dávkového zpracování:**

- **RJE (Remote Job Entry)**
  - dávka se připraví na jednom počítači (třeba na běžném PC)
  - a po síti se pošle ke zpracování na jiný počítač (např. superpočítač)
    - variantou je distribuovaná aplikaciální platforma
      - k dispozici je celý pool serverů, který si sám určí, na kterém serveru bude úloha zpracována



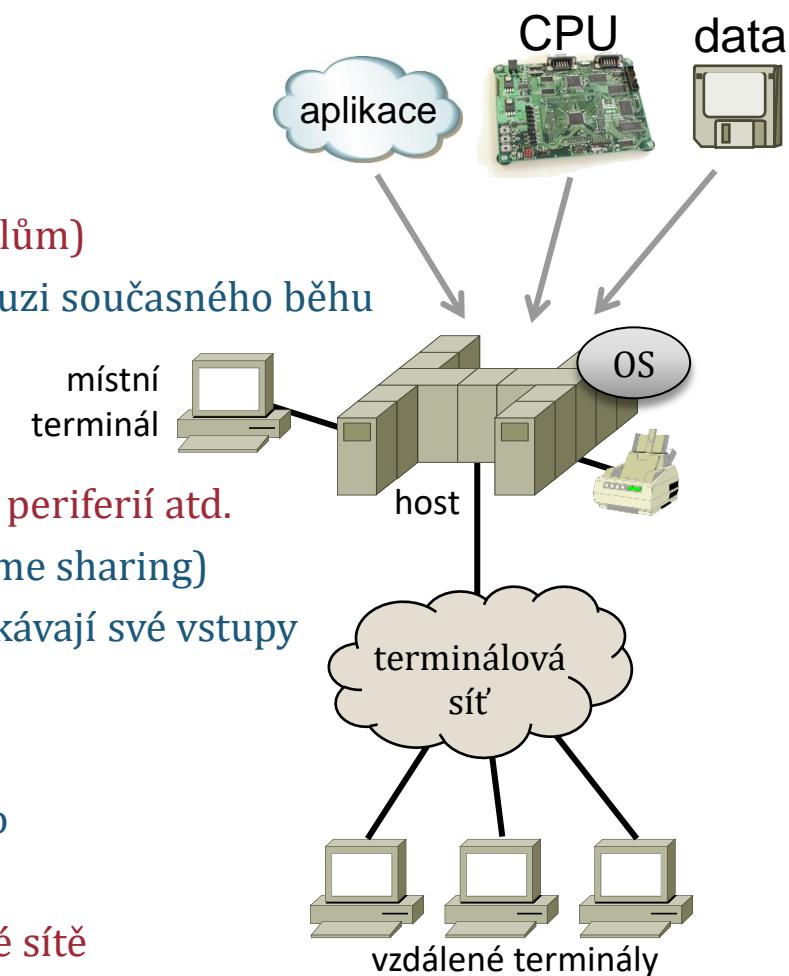
- **moderní obdoba**

- **model autonomních agentů**
  - autonomní agent je obdoba dávky
    - jde o celek (dávku) s určitým zadáním, který se při jeho plnění chová autonomně
      - obvykle: cestuje po síti a plní zadaný úkol, například vyhledávání informací, zdrojů apod.



# model host/terminál

- dávkové zpracování neumožňovalo přímý kontakt s aplikacemi
  - proto, že neexistovaly terminály (uživatelská pracoviště)
    - také proto, že HW byl málo výkonný, neexistoval SW umožňující současný běh více úloh
- změna nastala až když:
  - byly dostupné terminály
  - byl výkonnější HW
  - SW umožňoval sdílení času
    - současný běh více úloh („patřících“ různým uživatelům)
      - fakticky jde o jejich rychlé střídání, vytvářející iluzi současného běhu
- nový model: **host/terminál**
  - **host** (hostitelský počítač)
    - je „hostitelem“ zdrojů: dat a aplikací, paměti a CPU, periferií atd.
      - na host-u běží aplikace v režimu sdílení času (time sharing)
      - své výstupy posílají na terminál, z terminálu získávají své vstupy
  - **terminál**
    - je jednoduché vstupně/výstupní zařízení
      - například kombinace klávesnice a tiskárny, nebo
      - kombinace klávesnice a obrazovkového displeje
    - obvykle je více terminálů propojeno do terminálové sítě



# model host/terminál

The image shows two windows illustrating the host/terminal model:

- Telnet - anezka.felk.cvut.cz**: This window is a terminal emulator. It displays a menu from the "ON - LINE KATALOG ČVUT" system. The menu options are:
  - 1 Uvhledání titulu a KJ podle názvu
  - 2 Uvhledání titulu a KJ podle autora
  - 3 Uvhledání titulu a KJ podle klíčového slova
  - 4 Uvhledání titulů na pracovišti
  - 5 Novinky knihovny
  - 6 Exempláře čísel časopisu
  - 7 Help dotazovacího systému
  - 9 Ukončení činnosti
 Below the menu, there is a message in Czech: "Vaše volba, prosím" (Your choice, please) and instructions: "Vyberte volbu šipkami nebo číslem" (Select an option with arrows or number) and "Pro návrat do předchozího menu stiskte ESC" (To return to the previous menu, press ESC). The window title bar says "CZECH TECHNICAL UNIVERSITY IN PRAGUE (CVUT)" and "Serial # : 237229".
- TN3270 - earn to host earn.cvut.cz**: This window is a TN3270 emulator. It shows a virtual machine/system product screen with the following text:
 

VIRTUAL MACHINE/SYSTEM PRODUCT

```

      VVVVVV  VVVVVVVV  Virtual Machine/
      VVVVVV  VVVVVV    System Product
      VVV   VVV          Release 5.0
      0000000  VVV  VVV  cccccccc
      000000000  VVV  VVV  cccccccc
      00      00  VVV  VVV  CCC
      00      00  VVVVVV  CC
      00      00  VVVVVV  CCC
      000000000  VVVV   cccccccc
      0000000  VVV    cccccccc
      
```

Below this, it says "EARM Node for Czech Republic". At the bottom, it prompts the user to "Fill in your USERID and PASSWORD and press ENTER (Your password will not appear when you type it)". It also shows command-line inputs: "USERID ===> [REDACTED]", "PASSWORD ===> [REDACTED]", and "COMMAND ===>". The status bar at the bottom right shows "RUNNING CSEARN" and the timestamp "IBM-3278-2-E 11:38:38".

- výhoda:**
  - mezi hostitelským počítačem a terminálem se přenáší pouze kódy jednotlivých znaků
  - tj. malé objemy dat
- nevýhoda:**
  - uživatelské prostředí je pouze semigrafické !!!
  - tvořené pouze alfanumerickými znaky
  - nejde o (plně) grafické GUI
    - tak jak jej známe dnes

# vlastnosti modelu host/terminál

- vše je „na jedné hromadě“
  - všechny aplikace (úlohy) běží na hostitelském počítači
    - systémové úlohy, i úlohy všech uživatelů
  - 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
    - nikoli rastrová (bit-mapová) data grafického režimu, ale jen kódy alfanumerických znaků
  - vstupy z uživatelské klávesnice
    - jen kódy znaků, ještě žádná myš, trackball apod.
- terminály mohou být umístěny v různé vzdálenosti
  - blízko (místní, lokální terminály)
  - daleko (vzdálené terminály)
  - ..... (kdekoli v síti)
  - mohou být „skutečné“ i emulované
- výhody:
  - správa je jednodušší
  - vše stačí provést jen 1x, s dopadem na všechny uživatele
  - objem dat, přenášený mezi hostem a terminály, je malý
  - nároky na přenosovou síť jsou malé
  - mohlo to fungovat i v době, kdy sítě měly ještě nízké kapacity
  - terminály mohou být umístěny i ve velké vzdálenosti
  - terminál nemusí být „skutečný“ (jednoúčelový)
  - může být emulovaný
    - je to aplikace, běžící na běžném počítači

# vlastnosti modelu host/terminál

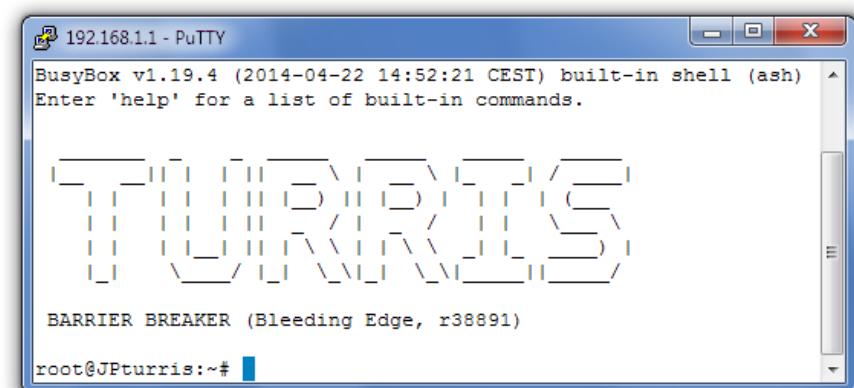
- **nevýhody:**

- pouze semigrafické prostředí
  - ve své době uživatelům tolik nevadilo
    - nebyli ještě tak „namlsaní“ dnešními GUI
- žádné polohovací zařízení
  - jako je myš či trackball
    - pouze posun o znakovou pozici pomocí kurzorových tlačítek na klávesnici
- uživatelé se cítí omezení tím, že dostupné zdroje sdílí s ostatními uživateli
  - i když je jim předkládána iluze, že mají počítač jen pro sebe
    - díky sdílení času a rychlému střídání úloh na procesoru



- **možnosti využití:**

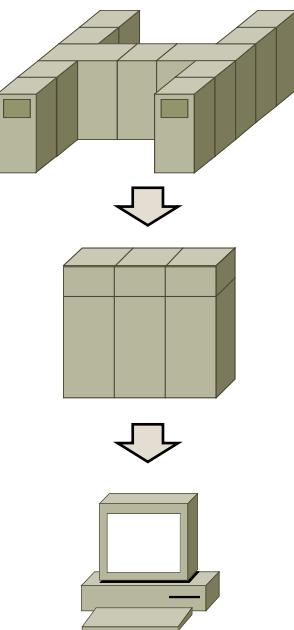
- v modelu host/terminál může být provozována jakákoli aplikace
  - jejíž výstupy (i vstupy) lze přesměrovat na (z) konkrétní terminál
    - nesmí zapisovat přímo do videoRAM
- aplikace si neuvědomuje, že funguje v prostředí sítě
  - a ani si to uvědomovat nemusí
    - přesto může být využívána v prostředí sítě
      - prostřednictvím vzdálených terminálových relací
- dodnes se využívá například pro vzdálenou správu různých zařízení



# model „desktop PC“

- **vývoj v oblasti HW:**

- počítače se stále zmenšovaly, zlevňovaly, rostl jejich výkon
  - velké sálové (střediskové) počítače (angl: mainframe) nahradily minipočítače
    - ale výpočetní model zůstával stále stejný: host/terminál
      - vše bylo centralizované („na jedné hromadě“) – na hostitelském počítači
      - zatímco terminály neměly žádnou vlastní výpočetní kapacitu – byla to jen pasivní V/V zařízení
  - stále se nevyplatilo (bylo příliš drahé) dát každému jeho vlastní počítač
    - přestože to uživatelé chtěli
  - teprve s postupem času (80. léta 20. století) se cena počítačů snížila natolik, že bylo možné dát každému počítač k jeho výhradnímu použití
    - jako **osobní počítač (Personal Computer, PC)**, na jeho stůl (**desktop**)



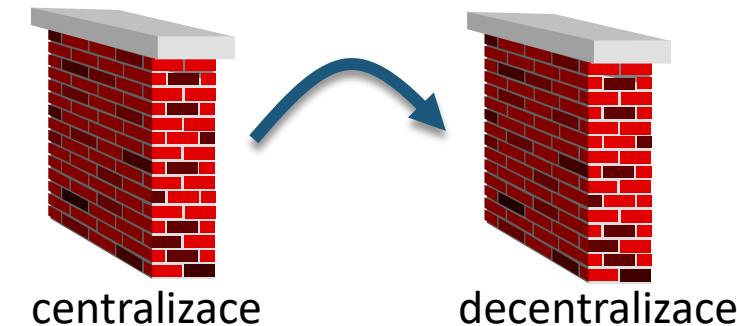
- **dochází ke změně výpočetního modelu**

- aplikace i data se stěhují
  - „z centra“ (z hostitelského počítače)
  - přímo k uživateli (na jeho osobní počítač)
- aplikace běží (a zpracovávají svá data)
  - na uživatelsově osobním počítači
  - aplikace si mohou myslet, že mají celý počítač jen pro sebe
    - mohou přistupovat přímo k paměti, ke všem periferiím,...
    - mohou např. zapisovat přímo do VideoRAM

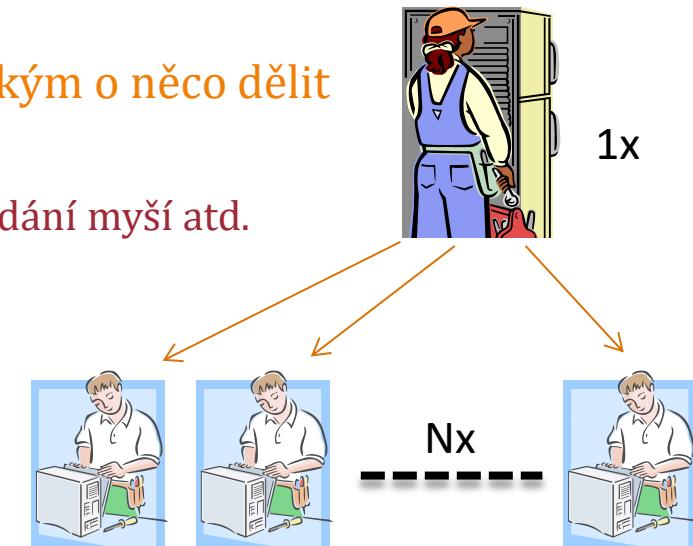


# model „desktop PC“

- přechod od modelu host/terminál k modelu „desktop“ byl zásadní změnou
  - skokem „ode zdi ke zdi“ (od jednoho extrému k jinému)
  - od úplné centralizace
    - kdy jsou všechny zdroje „na jedné hromadě“
      - centralizované na hostitelském počítači
  - k úplné decentralizaci
    - kdy jsou všechny zdroje distribuovány
      - nachází se přímo u svých vlastníků/uživatelů

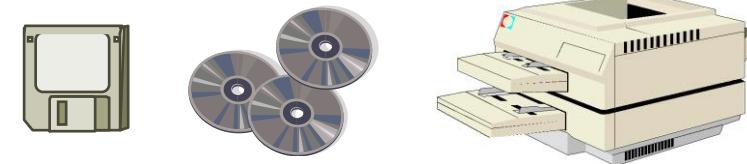


- řada problémů se tím vyřešila
  - uživatelé nemají (negativní) dojem, že se musí s někým o něco dělit
  - je možný vyšší komfort na uživatelském pracovišti
    - desktop aplikace již mohou mít plně grafické GUI, ovládání myší atd.
- ale vznikla řada zcela nových problémů
  - správa (desktop) aplikací a dalších zdrojů
    - dříve stačil 1x zásah v centru (na hostitel. počítači),
    - nyní nutno řešit Nx (na každém PC znovu)
  - replikace a sdílení zdrojů
    - jak to udělat, když uživatelé chtějí pracovat se stejnými daty? A každý je chce nějak měnit?
    - některé periferie jsou stále příliš drahé na to, aby je každý mohl mít jen pro sebe



# hledání zlaté střední cesty

- přechod „ode zdi ke zdi“ (od úplné centralizace k úplné decentralizaci) nebyl ideální
  - spíše přidělal více nových problémů, než kolik vyřešil těch „starých“
    - byl to přechod z jednoho extrému do jiného extrému
- lidé pochopili, že nejlepší je „zlatá střední cesta“
  - něco se vyplatí dát každému do výlučného použití, něco se vyplatí nadále sdílet



- co se vyplatí 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
    - které to jsou, nutno posuzovat individuálně

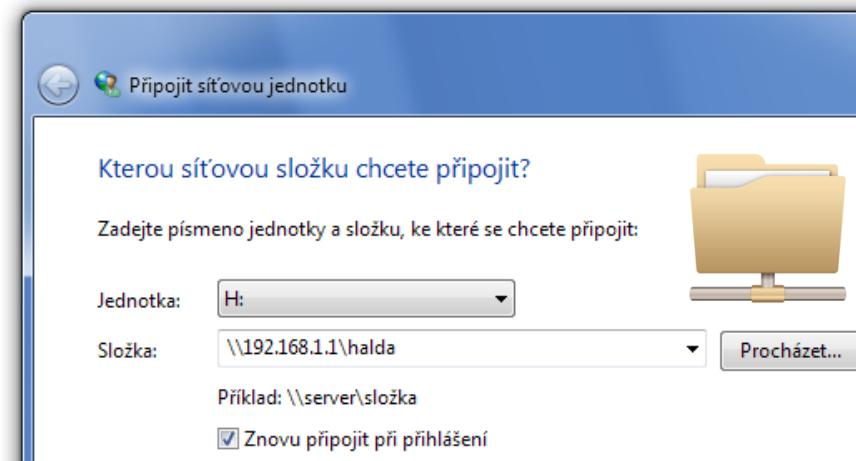
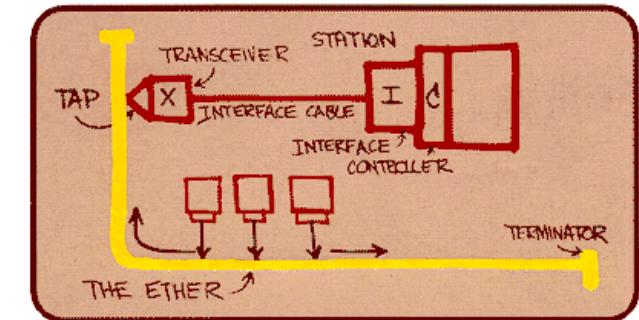
- co se vyplatí nadále sdílet?
  - drahé periferie
    - např. laserové tiskárny, modemy, .....
  - společná data
    - firemní databáze, sdílené dokumenty, .....
  - aplikace
    - vyžadující správné nakonfigurování a „údržbu“
    - .....

někdy se vyplatí centralizovat (uchovávat centrálně)  
třeba i soukromá data, kvůli jejich zálohování ....

# potřeba sdílení – motivace pro LAN

- pro možnost sdílení je nutné propojit jednotlivé osobní počítače do sítě
  - vzniká potřeba lokálních sítí (sítí LAN –Local Area Network)
- požadavky na LAN:
  - musí být dostatečně rychlé
    - aby se uživatelé dostali ke sdíleným zdrojům dostatečně rychle
    - aby nepoznali, že zdroj je umístěn „někde v síti“
      - a nikoli „u nich“ (přímo na jejich PC)
  - sdílení zdrojů v síti musí být neviditelné
    - pro aplikace i uživatele
      - aby uživatelé měli iluzi, že mají zdroj jen pro sebe
        - aby nevnímali jeho sdílení
        - aby nemuseli dělat nic jiného, než co by dělali při přístupu ke skutečně „místním“ zdrojům
      - aby aplikace nemusely vědět, že pracují v prostředí sítě, a přesto mohly přistupovat ke sdíleným zdrojům
        - **aby se daly využívat desktop aplikace, původně vyvinuté pro (desktop) PC, bez připojení do sítě**

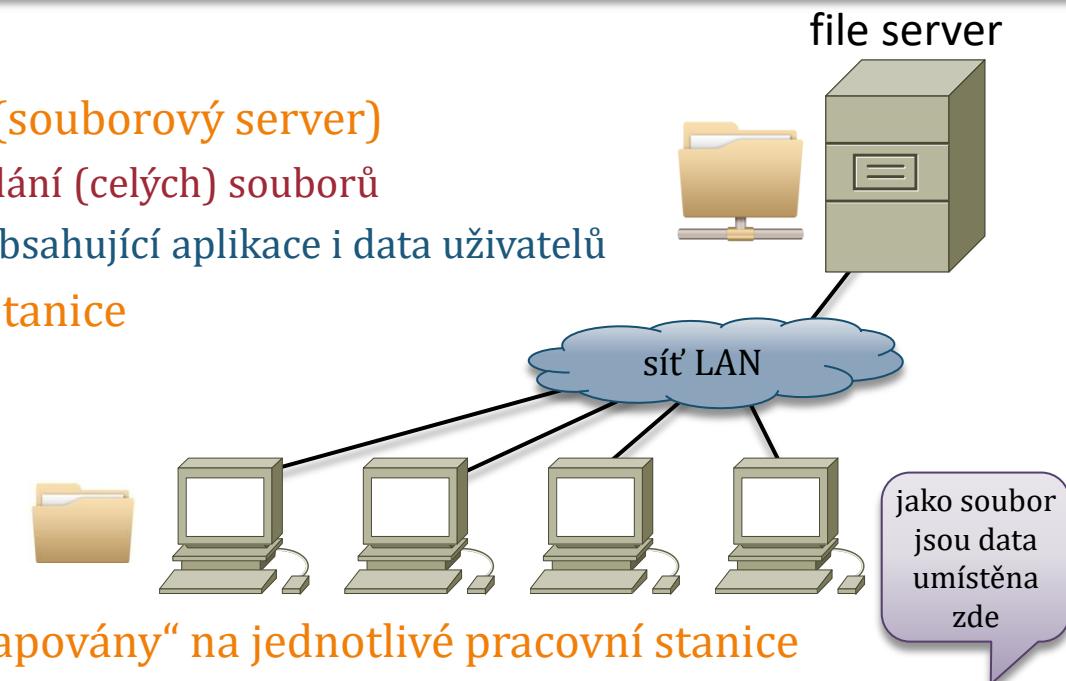
- k dispozici již je Ethernet
  - jeho 10 Mbit/s postačuje
  - v prostředí LAN, s krátkou RTT



# model file server/pracovní stanice

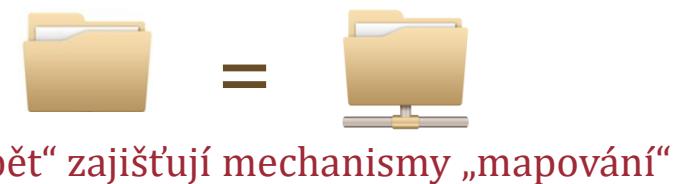
- **podstata modelu:**

- jeden uzel funguje jako file server (souborový server)
  - poskytuje službu, spočívající v ukládání (celých) souborů
    - jsou na něm umístěny soubory, obsahující aplikace i data uživatelů
- ostatní uzly fungují jako pracovní stanice
  - uživatelé na nich pracují:
    - spouští na nich své aplikace
    - zpracovávají na nich svá data



- **způsob fungování:**

- adresáře na file serveru jsou „namapovány“ na jednotlivé pracovní stanice
  - a chovají se jako místní adresáře
    - soubory, uložené na file serveru, se jeví jako „místní“ soubory
      - jejich sdílení není viditelné ani pro uživatele, ani pro aplikace
- spuštění aplikace je stejné, jako spuštění „místní“ aplikace
  - aplikace má formu souboru, který je umístěn na file serveru
    - ale jeví se jako místní
  - při spuštění aplikace je obsah souboru s aplikací přenesen na pracovní stanici a zde spuštěn
    - přenos zajišťují mechanismy „mapování“
- práce s daty je stejná, jako s „místními“ daty
  - data jsou v souboru na file serveru, jejich přenos „tam/zpět“ zajišťují mechanismy „mapování“



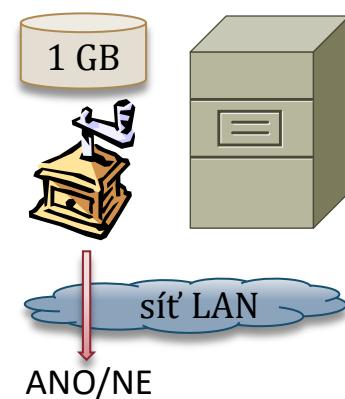
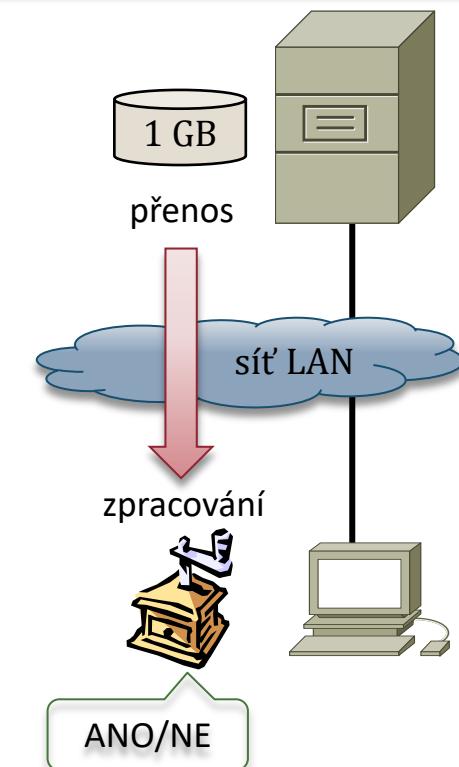
# model file server/pracovní stanice

- **výhody:**

- umožňuje sdílet aplikace i data
  - skrze jejich centrální umístění ve formě souborů
- aplikace nemusí tušit o existenci sítě
  - v tomto modelu lze provozovat i desktop aplikace

- **nevýhody:**

- problémy s konfigurací (jedné) aplikace pro více uživatelů
- problémy s přístupem více uživatelů ke stejným datům
  - například pokud chtějí měnit společný dokument
- data (často velká) mohou být přenášena po LAN zbytečně
  - mezi file serverem, kde se nachází, a
  - pracovní stanicí, kde jsou zpracovávána
    - extrémní příklad: v databázi o velikosti 1 GB má být nalezen výskyt nějakého řetězce
      - **tato databáze je umístěna (jako soubor) na file serveru**
      - **pro zpracování je 1 GB soubor přenesen na pracovní stanici**
        - **vlastní prohledávání probíhá na pracovní stanici**
        - **výsledkem je 1 bitová informace: ANO/NE**
          - **ale k jejímu získání se musel přenést 1 GB !!**



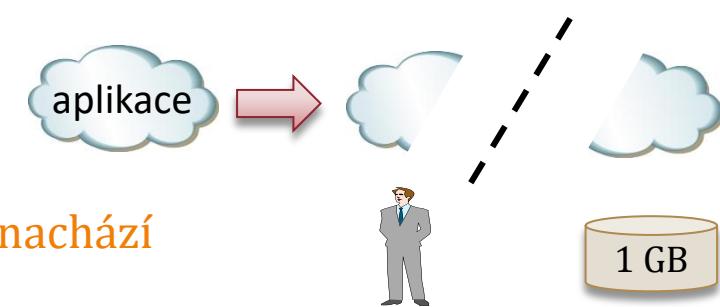
řešením je data nepřenášet, ale zpracovávat je přímo tam, kde se nachází

a přenášet pouze výsledek zpracování

# model klient/server

- **základní princip:**

- data se zpracovávají tam, kde se (data) nachází
  - obdobně pro jiné činnosti, než je zpracování dat
- s uživatelem se komunikuje tam, kde se (uživatel) nachází

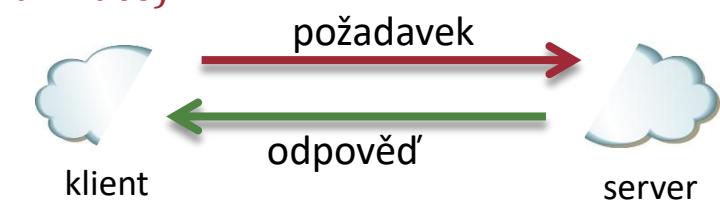


- **důsledek:**

- původně „jednolité“ (monolitické) aplikace se musí rozdělit na 2 části:
  - **server** (serverovou část): „běží“ tam, kde jsou data
  - **klient** (klientskou část): „běží“ tam, kde je uživatel
- rozhraní („řez“) mezi oběma částmi by mělo být voleno tak, aby jejich vzájemná komunikace byla co nejmenší
  - co do objemu přenášených dat (ne nutně četnosti komunikace)
    - aby zátěž přenosové sítě byla co nejmenší !!!
- aplikace si již musí uvědomovat existenci sítě

- **princip fungování modelu klient/server:**

- komunikace mezi oběma částmi aplikace je charakteru požadavek-odpověď
- server je pasivní a čeká, až dostane od klienta nějaký požadavek
  - a ten vyřídí – vygeneruje odpověď
- klient je aktivní: na základě aktivit uživatele generuje požadavky vůči serveru
  - a zprostředkovává poskytnutí odpovědi uživateli (prezentaci výsledků)



# model klient/server

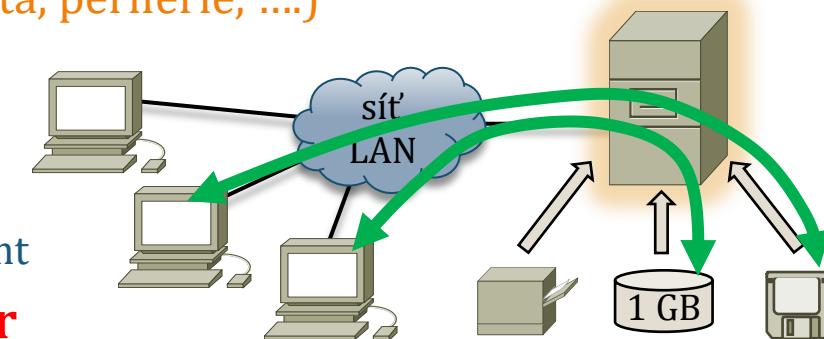
- v **modelu klient/server dnes funguje většina služeb v Internetu, např.**
  - World Wide Web, elektronická pošta, přenos souborů, .....
- **fungování modelu vyžaduje, aby existovala a dodržovala se konvence o:**
  - způsobu vzájemné komunikace mezi klientem a serverem
    - WWW: protokol HTTP
      - HyperText Transfer Protocol
    - el. pošta: protokol SMTP
      - Simple Mail Transfer Protocol
    - přenos souborů: protokol FTP
      - File Transfer Protocol
    - přenos souborů: protokol SMB
      - Server Message Block
    - další: DNS, DHCP, IRC, SIP, SNMP, NTP, .....
  - formátu a významu toho, co se mezi serverem a klientem přenáší
    - WWW: jazyk HTML
      - HyperText Markup Language
    - el. pošta: formát zpráv
      - standard RFC 822 .....
    - přenos souborů: formát souborů
      - vlastní konvence
    - .....
- **(ne)výhody modelu klient/server:**
  - všechny zdroje jsou centralizované
    - jsou umístěny na serveru
  - každá aplikace má svého klienta
    - je nutné ho udržovat (zajišťovat jeho správu), učit uživatele používat klienta i celou službu



# alternativa: model peer-to-peer

- **model klient/server je centralizovaný**

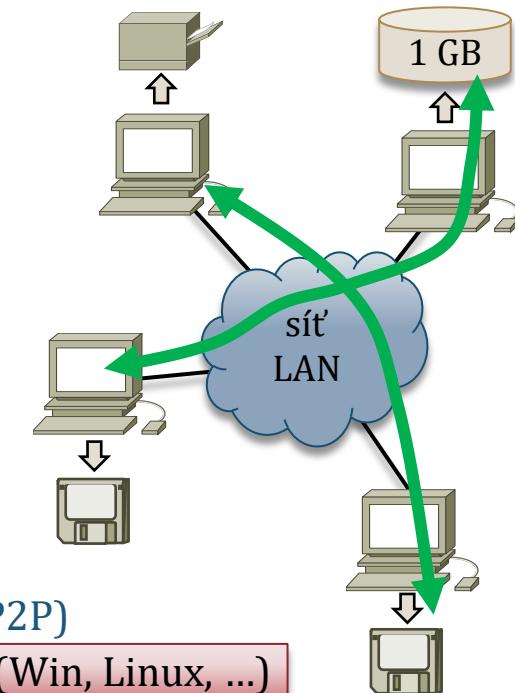
- předpokládá a očekává, že všechny zdroje (data, periferie, ....) se přesunou na centrální server
- je to asymetrické řešení
  - zavádí nerovnost mezi uzly
    - některý je (pouze) server, jiný je (pouze) klient



- **je možná alternativa: model peer-to-peer**

- zdroje zůstanou „tam, kde jsou“
  - kde vznikají, u svého vlastníka, ....
    - na konkrétních uzlech v (lokální) síti
- každý uzel se chová současně:
  - jako server
    - zpřístupňuje ostatním ty zdroje, které má u sebe
  - jako klient
    - žádá o zdroj „toho, kdo zdroj má“ (server)
- je to symetrické řešení
  - všechny uzly si jsou (mohou být) rovny
  - v angličtině jsou sobě rovné uzly označovány jako „peers“
    - proto je tento model označován jako peer-to-peer (zkratkou P2P)

podpora peer-to-peer sítí je dnes (nativně) dostupná ve většině OS (Win, Linux, ...)



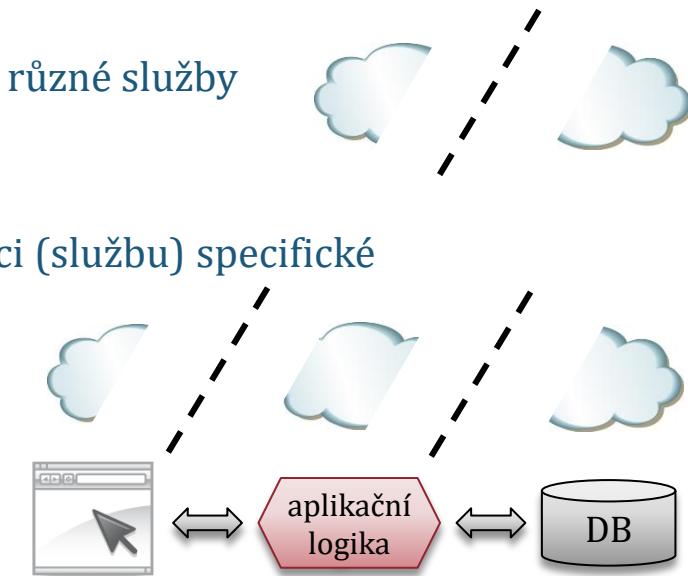
# 3 úrovňový model klient/server

- **připomenutí:**

- nevýhodou modelu klient/server je specifičnost klienta (klientské části aplikace)
  - každá služba má svého vlastního klienta = složité a drahé (na správu, údržbu, ....)

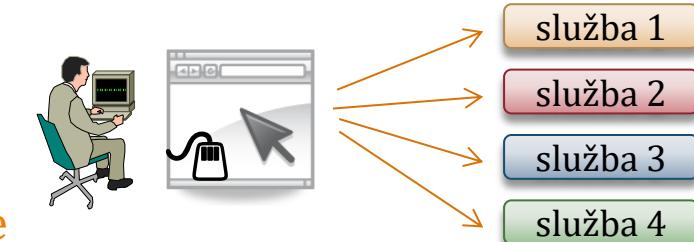
- **řešení:**

- rozdělit původně jednotlivou aplikaci nikoli na 2 části (klient/server), ale na 3 části
  - prezentační část
    - která může být univerzální = stejná (společná) pro různé služby
      - v praxi obvykle: webový prohlížeč (browser)
  - aplikační část
    - ve které je soustředěno vše, co je pro danou aplikaci (službu) specifické
      - je zde implementována tzv. aplikační logika
  - databázovou část
    - ve které jsou uchovávána data
      - může být zcela standardní = jakákoli databáze



- **výhody:**

- nižší náklady na správu a údržbu na straně uživatelů, jednodušší používání, ....
  - když je klientská část společná pro více služeb
- nižší náklady na implementaci a provoz
  - je použita zcela standardní databáze
- jednotlivé části mohou být libovolně daleko od sebe



# desktop vs. network-centric computing

- jde o dva rozdílné pohledy na celkovou architekturu počítačů, výpočetní modely i roli sítí

- desktop computing

- „vše je na desktopu“ (na uživatelském počítači)
  - aplikace jsou připraveny (nainstalovány) a provozovány na desktopu
    - nebo alespoň jejich klientské části
  - další zdroje (data, periferie) jsou také na desktopu
    - nebo alespoň „namapovány“ jako místní
- síť slouží pouze k přenosům a komunikaci
  - ale není „zdrojem všeho“
- důsledek:
  - možnosti na různých počítačích (desktopech) jsou různé
    - *„co jde dělat na jednom počítači, nemusí jít dělat na ostatních .....“*
      - nemusí zde být dostupné stejné aplikace
      - nemusí zde být dostupná stejná data



- network-centric computing

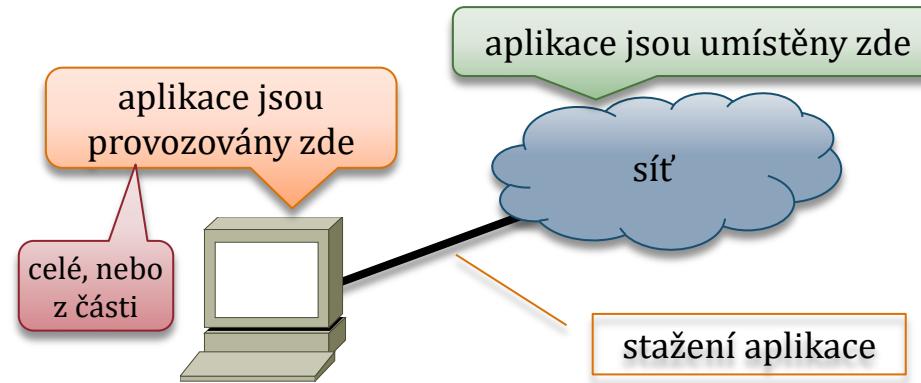
- „vše je v síti“
  - aplikace, data i další zdroje se nachází v síti
    - ale stahují se na uživatelský počítač, nebo používají na dálku
- síť je „primární“ a hlavní
  - uživatelské počítače jsou spíše jen „koncová zařízení“
    - nemusí jít o klasická PC, mohou to být i zařízení jiných typů
    - dnes např. tablety, mobily, ...
- důsledek (efekt):
  - možnosti na různých počítačích mohou být stejné
    - *„co jde dělat na jednom koncovém zařízení, jde dělat i na ostatních .....“*
- cílem je:
  - snížit náklady na správu a údržbu
    - TCO, Total Cost of Ownership

patří sem i model klient/server

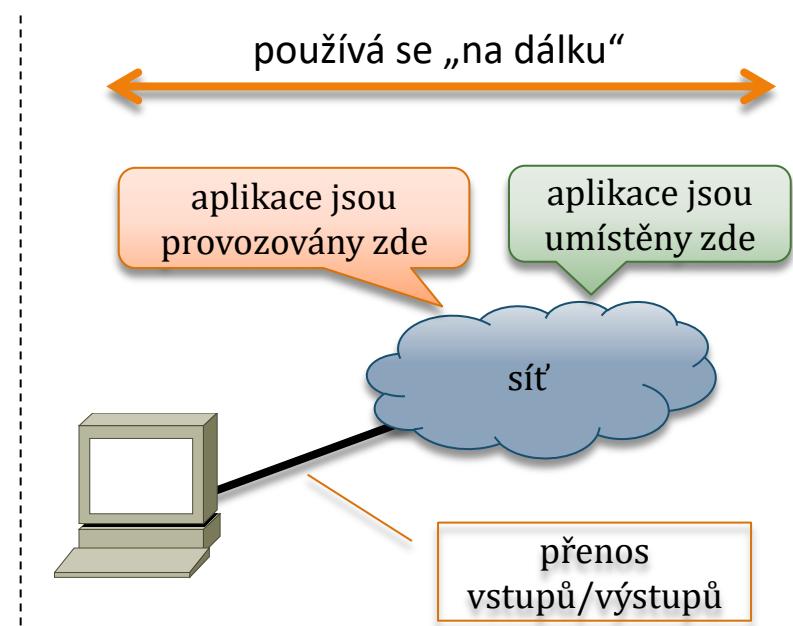
# network-centric computing

- **jde o obecný koncept**

- může být naplněn (realizován) různými způsoby, s využitím různých výpočetních modelů



- aplikace („spustitelný kód“) jsou umístěny v síti, ale jsou stahovány do koncového zařízení, a spouštěny a provozovány v něm
- takto funguje například:
  - tenký klient (též: NC, Network Computer)
  - webové aplikace
  - tzv. „bohaté internetové aplikace“
    - RIA, Rich Internet Applications
  - tzv. streaming aplikací
- po použití je spustitelný kód jednoduše zahozen
  - a příště stažen znova „stahuje se“



- aplikace jsou umístěny, spouštěny a provozovány v síti, do koncového zařízení se přenáší pouze jejich výstupy
- takto fungují například:
  - aplikační servery
  - výpočetní model server-based computing

# tenký a tlustý klient, PC vs. NC

- **klasický osobní počítač (PC):**
  - je dimenzován tak, aby byl předem připraven na možné požadavky uživatele
  - jsou na něm dopředu instalovány aplikace (celé nebo jejich klientské části), pro případ že by je uživatel potřeboval
    - podle toho je dimenzován výkon CPU, velikost RAM a HD, periferie .... i OS
  - **výhoda**
    - když uživatel něco skutečně chce, dosáhne toho rychle
    - rovnou si spustí tu (připravenou) aplikaci, která splní jeho požadavek
      - například na editaci textu, práci s el. poštou, brouzdání webem, ....
  - **nevýhoda**
    - počítač musí být hodně výkonný (=dražší)
    - počítač musí být dopředu „zabydlen“
      - vše nainstalované, nakonfigurované ...
    - náklady na správu a údržbu jsou vysoké

též: tzv. **tlustý klient (PC)**

- tzv. **tenký klient**



- je dimenzován minimalisticky
  - nejsou na něm nainstalovány žádné aplikace
  - aplikace (kód) se stahují (ze sítě) až v okamžiku jejich skutečné potřeby
    - na základě požadavku uživatele
  - po použití se stažený kód zahodí
- počítač má jen jednoduchý OS
  - schopný přjmout požadavek uživatele a stáhnout vše potřebné
- **výhoda**
  - nižší nároky na HW
  - a zcela jiné nároky na SW
  - velmi jednoduchá (levná) správa
  - „blbovzdorné“ zařízení
- **nevýhoda**
  - všechny aplikace je nutné napsat znova
  - stažení aplikací může trvat (příliš) dlouho



též: tzv. **NC (Network Computer)**



# osud tenkých klientů



- původní představa (**motivace**), cca 1995-1997:
  - půjde o náhradu klasických PC („tlustých klientů“)
    - spíše o vytlačení dua Microsoft+Intel (konkurence: Oracle+Netscape+Sun+IBM)
  - aplikace, stahované ze sítě, budou psány v Javě, jako applety
    - a koncové zařízení („tenký klient“) bude fungovat jako Java Virtual Machine
- **problémy:**
  - bylo nutné přepsat všechny aplikace do podoby Java appletů
    - např. firma Corel vytvořila v Javě celý kancelářský balík (Corel Office for Java)
  - ukázalo se, že stahování appletů ze sítě (i dostatečně rychlé) je příliš pomalé na to, aby na to uživatel vydržel čekat
    - když např. řekne, že chce psát text, stahuje se celý textový editor v podobě appletu
      - a než se stáhne, zavede do paměti a spustí, trvá to z pohledu uživatele neúnosně dlouho
- **důsledek**
  - tenký klient (NC) se v praxi moc neujal
    - jeho nevýhody převážily nad výhodami
      - měl být levnější, ale i klasická PC („tlustý klient“) zlevnila a snížila své TCO
      - tvorba aplikací pro tenké klienty (NC) byla problematická
  - dnes se NC využívá jen pro specifické účely
    - pro jednoúčelové využití (např. na bankovní přepážce), kde se neprovozují jiné aplikace

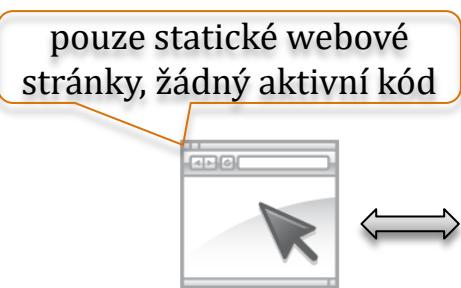


# webové aplikace

- jde o hodně široký pojem, s mnoha různými výklady

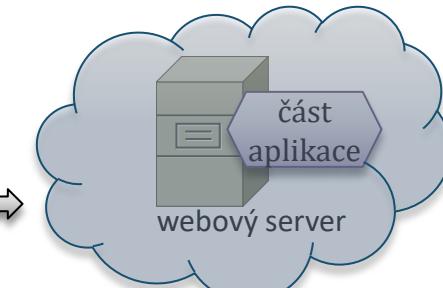
- starší (historické) pojetí:

- celá aplikace běží v síti
- uživatel pracuje s běžným webovým browserem, který mu zobrazuje statické webové stránky
- statické stránky generuje webový server, běžící v síti
  - za ním je „schována“ samotná aplikace
  - aplikační logika (a ev. i databáze)
- vstupy od uživatele se zadávají jako data do formulářů (v rámci webových stránek)
  - a odesílají na server
  - metodami GET či POST
- podobné 3-úrovňovému modelu klient/server
  - zasazenému do prostředí webu a Internetu



- modernější pojetí:

- „něco“ (nějaký kód) se stahuje a běží uvnitř browseru, se kterým pracuje uživatel
  - typicky: v browseru běží část aplikace
    - různě velká část, případně i celá aplikace
    - například: část aplikace, zajišťují „zobrazovací logiku“ (GUI logic)
  - „zbytek“ aplikace běží v síti (na serveru)
    - aplikační logika, data, ....
- browser se chová jako univerzální (softwarový) tenký klient
  - výkonný kód se do něj stahuje, použije a pak zahodí (smaže)
  - možnosti (formy kódu v browseru):
    - JavaScript, Java, DHTML, .....



# příklad: jízdní řády

- vyhledávání dopravního spojení

- webová aplikace (spíše) ve starším pojetí
  - dotazy se zadávají do políček formulářů a odesílají na server (stisknutím tlačítka)
  - vlastní funkčnost na straně klienta: našeptávač
    - realizováno pomocí Javascriptu

- odpověď generuje server v síti
  - aplikace vyhledá spojení
  - webový server převede nalezené spojení do podoby HTML stránky a předá klientovi (browseru) k zobrazení

IDOS - Vlaky - Vyhledání sp...

jizdnirady.idnes.cz/vlaky/spojeni/

**jizdnirady.cz**

SPOJENÍ ODJEZDY ZASTÁVKOVÉ JŘ SPOJE MAPA

Jízdní řád: ② ③

Odkud: Praha

Kam: Hustopeče

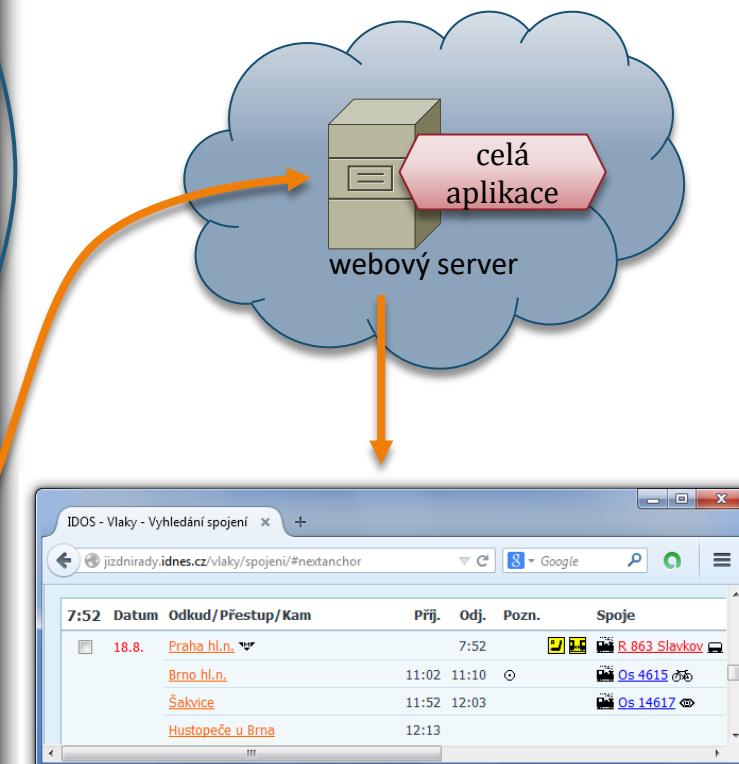
Hustopeče obec, okres Břeclav

Hustopeče nad Bečvou obec, okres Přerov

Hustopeče n. Bečvou stanice (vlak), okres Přerov

Hustopeče u Brna stanice (vlak), okres Břeclav

HLEDAT Rozšířené zadání Výchozí



# další vývoj webových aplikací

- **vývoj „u uživatele“**

- **původně:**

- „*kód, který se stahuje ze sítě*“, běží v browseru nativně
      - bez nutnosti nějakých doplňků (plug-inů) – např. JavaScript, DHTML
        - resp. podpora tohoto typu kódu je v browseru zabudována již od jeho výrobce

- **nově:**

- „*kód, který se stahuje ze sítě*“, vyžaduje ke svému běhu v browseru dodatečnou podporu
      - je nutné instalovat do browseru odpovídající doplněk (ve formě plug-inu)
        - platí např. pro Java (vyžaduje JVM), Flash/Flex, Adobe Air, Silverlight, ...
          - nejčastěji je plug-in od nějaké 3. strany (ne od výrobce browseru)
      - jde vlastně o novou a samostatnou platformu, na které běží „*kód, který se stahuje ze sítě*“
        - tato platforma může být vytvořena uvnitř browseru
          - tzv. **in-browser**: kód běží uvnitř browseru
        - nebo může být vytvořena samostatně, nezávisle na browseru
          - tzv. **out-of-browser**: kód běží mimo browser, v rámci samostatné aplikace, která vytváří potřebné prostředí/platformu (canvas, sandbox, ....)

- **nejnověji (opět):**

- „*kód, který se stahuje ze sítě*“, běží v browseru nativně
      - díky otevřeným standardům, jako je HTML5

HTML+kód



plug-in



1.



2.



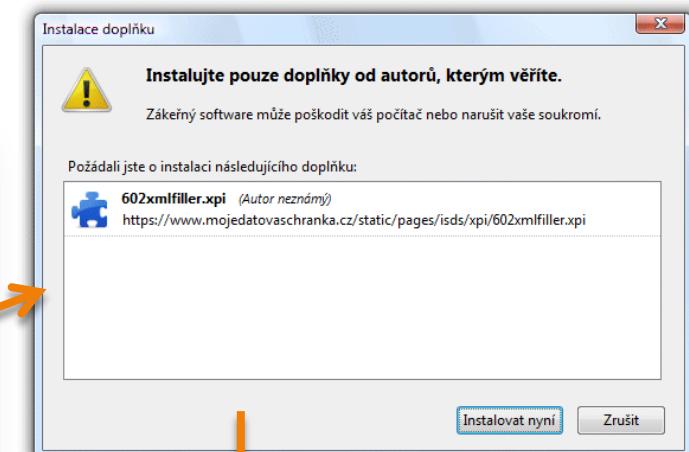
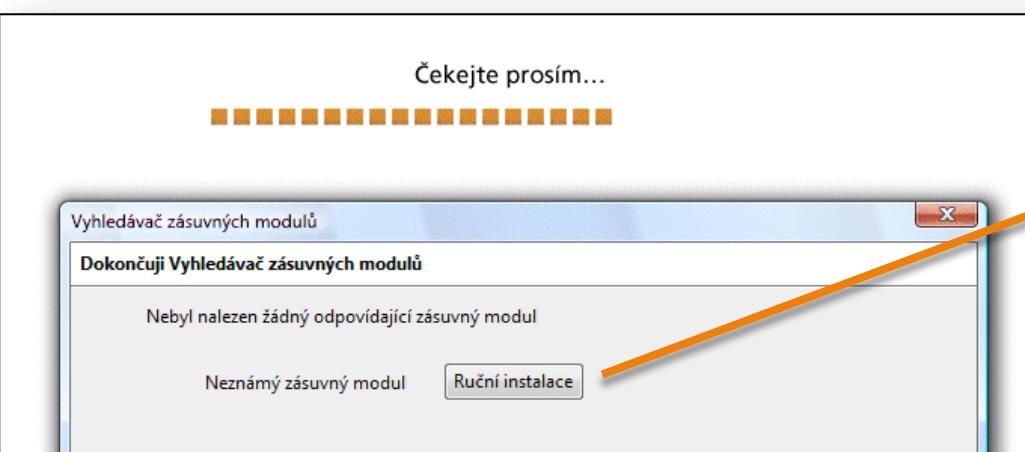
nebo:



„out-of-browser“

# příklad: datové schránky

- **datové zprávy, přenášené skrze datové schránky (ISDS) mají formát ZFO**
  - jde o proprietární formát, který žádný browser nativně („sám od sebe“) nepodporuje
- **původní řešení:**
  - server ISDS (Informačního systému datových schránek) nechával zobrazení obsahu datové zprávy (ve formátu ZFO) na klientovi (na browseru)
  - což vyžadovalo instalovat podporu formátu ZFO
    - doplněk 602 XML Filler od Software602



Stav	Doručeno	Odesíatel	Věc	Naše spisová zn. a č.j.	Váše spisová zn. a č.j.
	30.7.2009 10:48:07	Kancelář veřejného ochránce práv Udolní 39, 60200 Brno-střed, Česká Republika	Odpovídá na dotaz novináře na statistiku využití DS	ČJ-PDCJ/2063/2009	

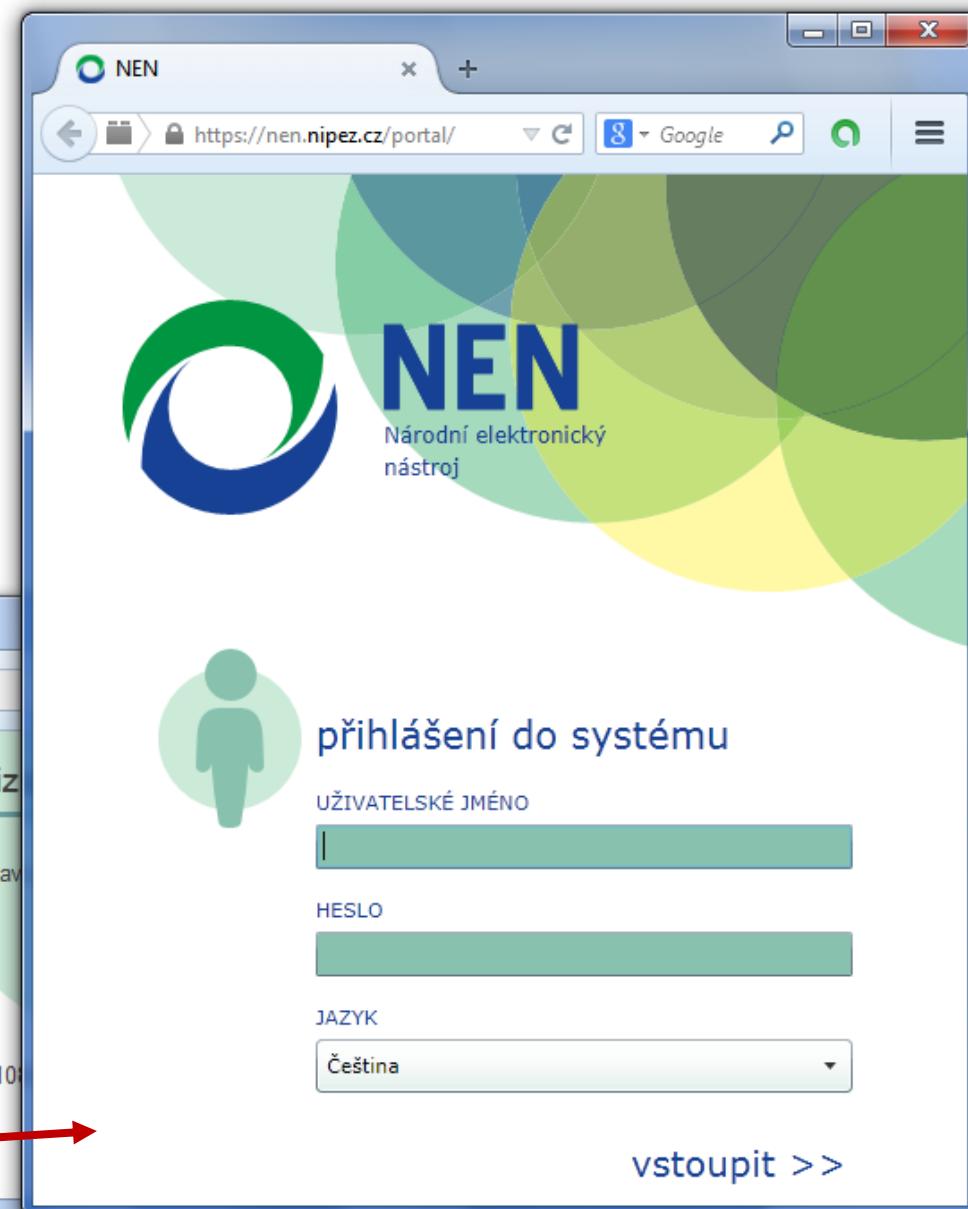
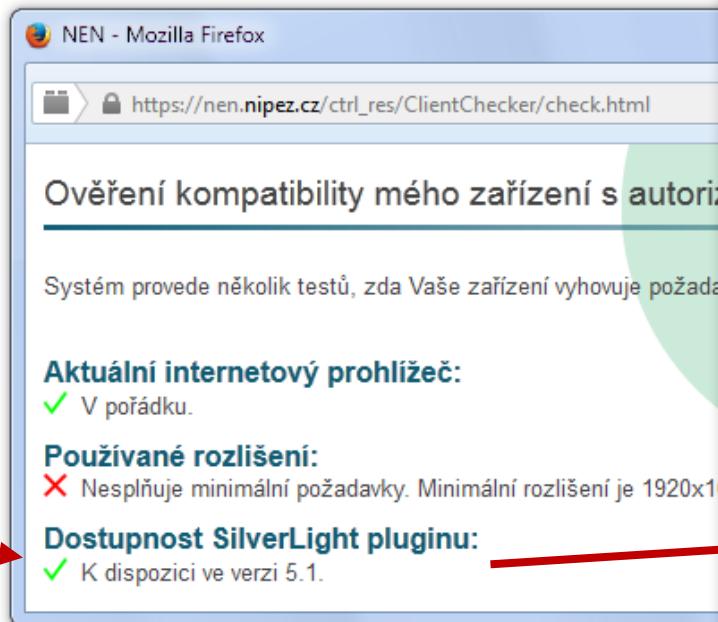
Správce: Ministerstvo vnitra České republiky, Provozovatel: Česká pošta, s.p.

Ochrana osobních údajů | Prohlášení o plnouspornosti

- **dnešní řešení (od 30. 7. 2012):**
  - formát ZFO již „rozbalouje“ server
    - klientovi předává jeho obsah již převedený do HTM

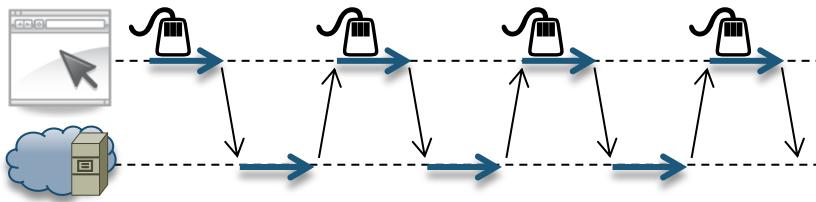
# příklad: NEN

- **NEN: národní elektronický nástroj**
  - platforma pro veřejné zakázky
- **v browseru běží celá aplikace NEN**
  - je napsána v Silverlight-u
    - technologie od společnosti Microsoft
  - vyžaduje instalaci plug-inu
    - s podporou technologie Silverlight



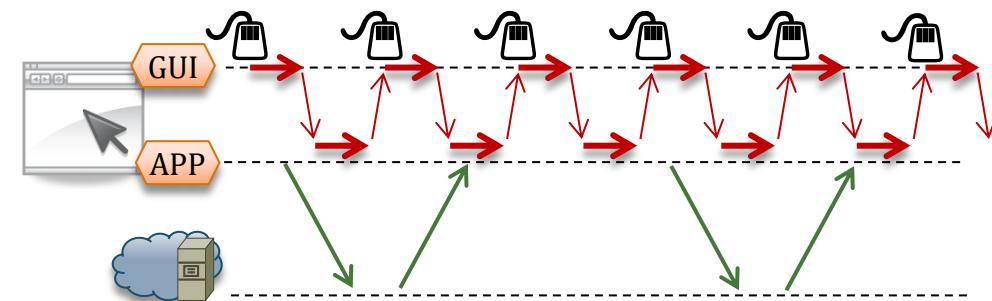
# další vývoj webových aplikací

- týká se dění „mezi uživatelem a sítí/serverem“
  - jde o přechod ze synchronní komunikace na asynchronní komunikaci
- **synchronní webové aplikace**
  - komunikace mezi browserem a sítí/serverem probíhá na základě přímých podnětů uživatele
    - synchronně s aktivitami uživatele
      - tj. nedělá se nic „dopředu“, nezávisle na uživateli a jeho aktivitách
    - komunikace je vždy „plná“, například:
      - pokud komunikace probíhá na úrovni načítání webových stránek:
        - dojde k načtení celé nové stránky
          - nikoli jen nějaké dílčí části
          - až když uživatel na něco klikne
      - využívají se standardní „prostředky“
        - HTTP, HTML, CSS, ....



## • asynchronní webové aplikace

- komunikace mezi browserem a sítí/serverem může probíhat „na pozadí“
  - asynchronně s aktivitami uživatele
    - browser může načítat nová data „dopředu“, nezávisle na uživateli a beze změny zobrazení
  - komunikace může být „částečná“
    - browser si může vyžádat třeba jen dílčí část dat, kterou potřebuje
      - když předvírá aktivity uživatele
  - jsou nutné nové druhy „prostředků“



# příklad: AJAX a Google maps

- Google Maps je „asynchronní“ webová aplikace

- využívá technologii AJAX

- pomocí které si stahuje jen ta (částečná) data, která potřebuje
    - nemusí se načítat celá nová webová stránka
  - pomocí které si data stahuje tehdy, kdy sám uzná za vhodné
    - nemusí čekat na aktivity/požadavky uživatele

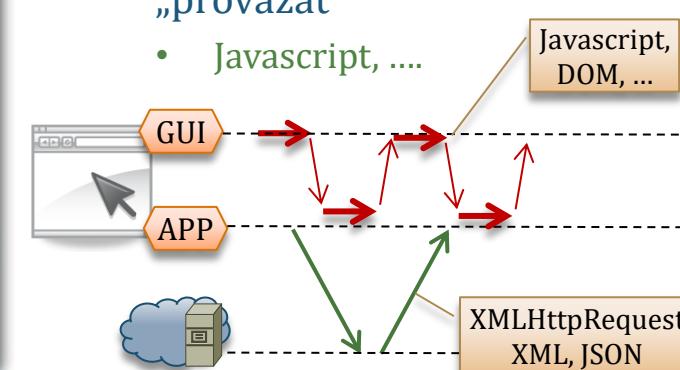
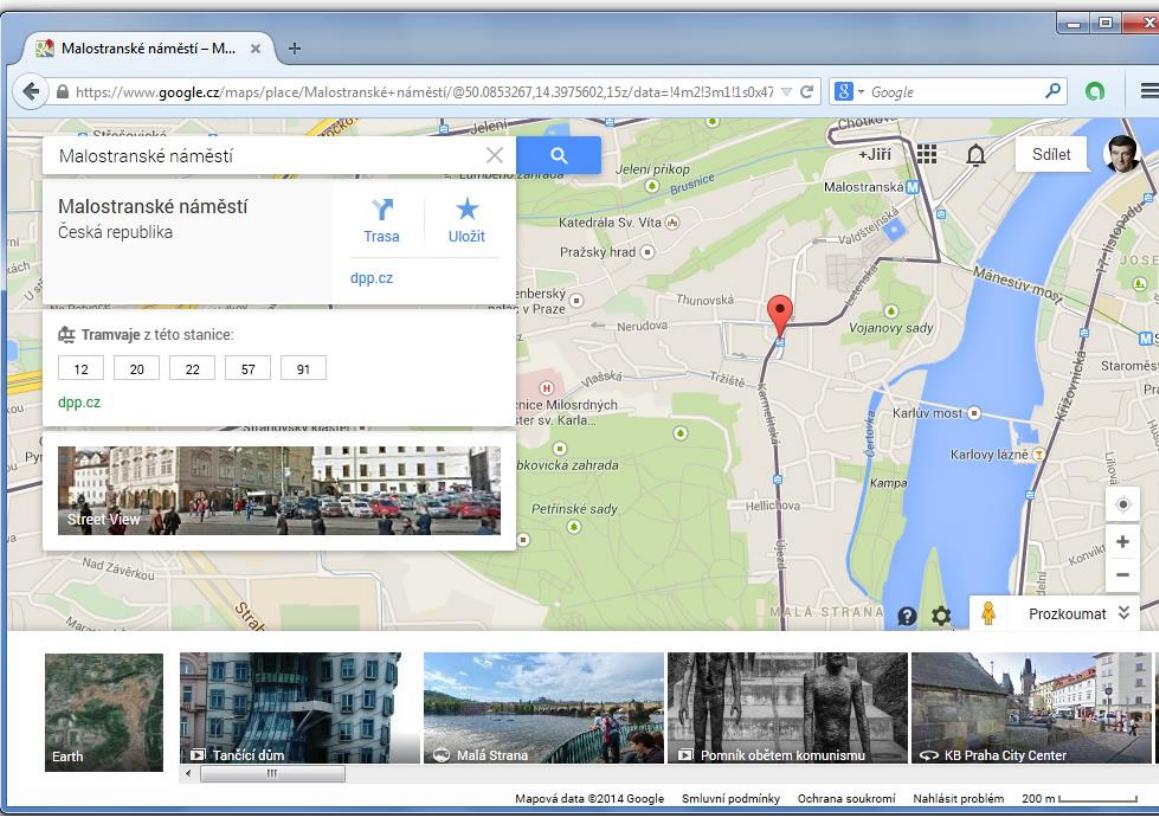
Asynchronous Javascript and XML

- zahrnuje:

- nový prostředek pro vznášení „dílčích požadavků“
      - objekt XMLHttpRequest
    - vhodné formáty dat pro (dílčí) přenos
      - XML, JSON ....

- dále:

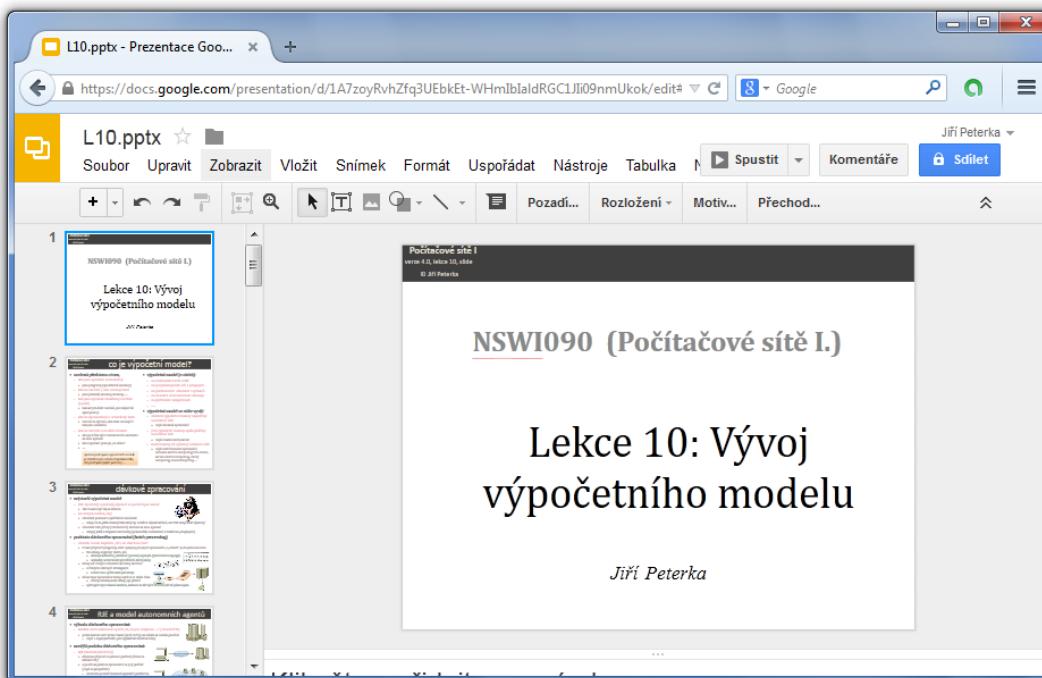
- možnost „zasahovat“ do dílčích částí HTML stránek
    - DOM (Document Object Model), ....
  - možnost vše vhodně „provázat“
    - Javascript, ....



# bohaté internetové aplikace

## • RIA: Rich Internet Applications

- jde o „lepší“ variantu webových aplikací
  - někdy považovanou za samostatnou kategorii
- klientská část („kód, který se stahuje ze sítě“) se svými schopnostmi i uživatelským komfortem vyrovná samostatné (desktop) aplikaci
  - nebo ji dokonce předčí
  - zjednodušeně: uživatel nevnímá rozdíl mezi desktop aplikací a RIA aplikací



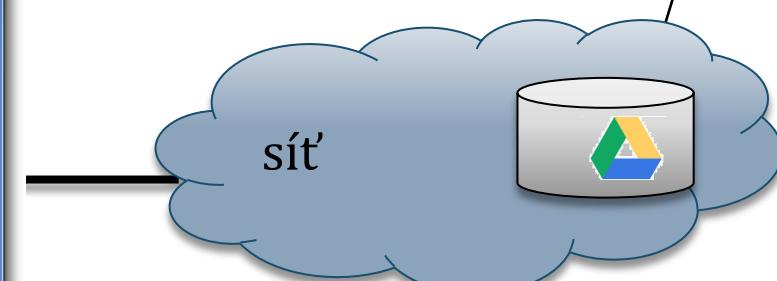
## • obvyklé řešení:

- klientská část zajišťuje většinu činností, spojených s fungováním aplikace
- serverová část uchovává data a stavové informace

## • příklady

### – Google Docs

- klientská část zajišťuje vlastní práci s dokumenty
  - například všechny funkce editoru realizuje klientská část
- serverová část uchovává jednotlivé dokumenty uživatele
  - plus veškeré nastavení



# výhody webových/RIA aplikací

- mají výhody (i nevýhody), plynoucí z modelu network-centric computing
- výhody
  - obecně: běží všude a stejně
    - na všech počítačích a podobných zařízeních
    - nejčastěji v rámci browseru (in-browser), případně samostatně (out-of-browser)
  - nemusí se instalovat ani aktualizovat
    - na uživatelských zařízeních
    - protože se stahují ze sítě
  - mají nižší náklady na správu a údržbu
    - stačí vše řešit centrálně (v síti / na serveru)
- připomenutí:
  - jde o jednu z možností, jak realizovat výpočetní model „network-centric computing“
    - taková, kdy část „funkčnosti“ aplikace (nebo i celá) je vykonávána na počítači uživatele
  - jde o určitou renesanci (vylepšenou verzi) tenkého klienta
    - kdy se do koncového zařízení může stahovat celá aplikace, nebo jen její část
    - zbytek aplikace je v síti (na vhodném serveru)
    - kdy roli NC obvykle hraje webový browser (varianta „in-browser“)
    - ale může to být i nějaký samostatný program, vytvářející vhodné prostředí



ADOBE® AIR™

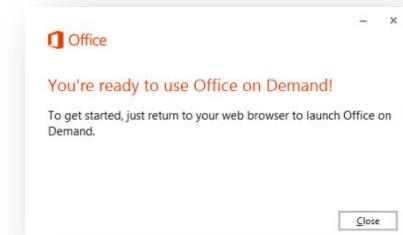
# tzv. streaming aplikací

- **stejná myšlenka jako u NC/tenkého klienta:**

- stáhnout aplikaci (a spustit ji) až na základě potřeby (žádosti) uživatele
  - stahuje se „ze sítě“, z vhodného serveru, kde musí být dopředu připravena, ve správné podobě

- **realizace:**

- je intelligentnější než u NC
  - předpokládá, že uživatel nepotřebuje plnou funkčnost aplikace
    - ale v každém okamžiku využívá vždy jen nějakou její část
- stahování je progresivní: aplikace se stahuje po částech
  - nejprve se stáhne „tak malá část“, aby se dalo začít
    - aby se uživateli něco spustilo a on měl s čím komunikovat
    - aby odezva byla co nejrychlejší, aby uživatel dlouho nečekal
  - další části se stahují „na pozadí“, během práce uživatele, budťo:
    - proaktivně, v očekávání, že příslušná část bude zapotřebí
    - reaktivně, jako reakce na konkrétní požadavek uživatele
- po použití (skončení práce s aplikací) se stažený kód zahodí
  - nebo se uchová v cache paměti, aby se příště „stáhl“ rychleji
    - aplikace se na počítači neinstaluje !!



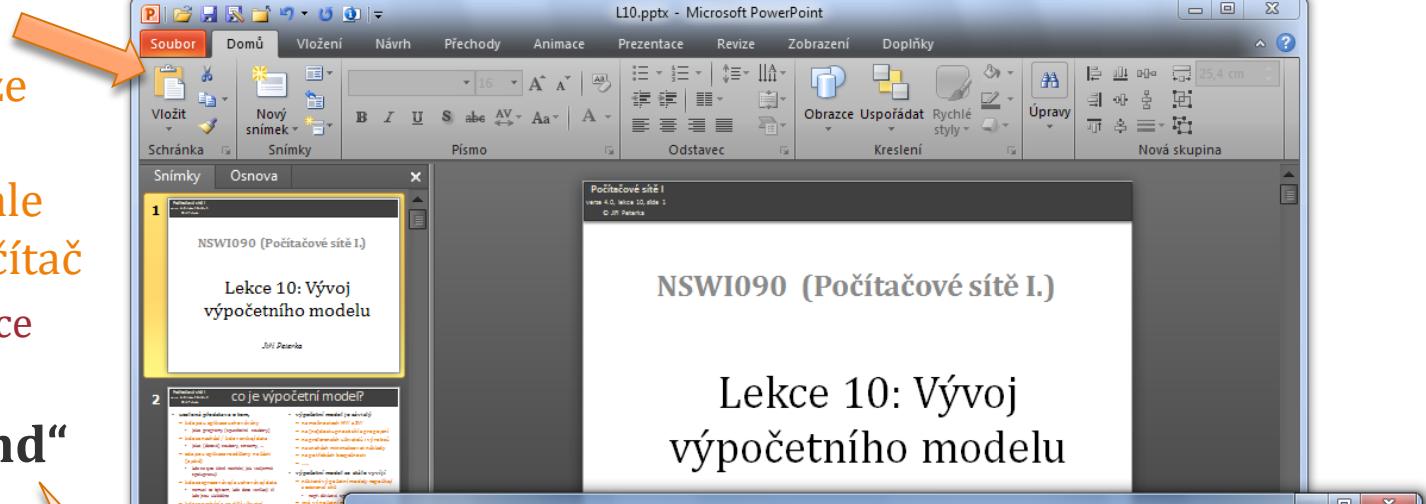
- **příklad:**

- používá se například u MS Office 365 (Office on Demand, „Click to Run“)
  - výsledný efekt je pro uživatele stejný, jako při použití „lokálně nainstalované“ aplikace

# příklad: MS Office 365

## • desktop verze

- monolitická verze aplikace, lze nainstalovat trvale na konkrétní počítač
  - v rámci MS Office 2013 i 365



## • verze „on demand“

- monolitická verze, stejné schopnosti jako desktop verze, ale neinstaluje se „natrvalo“
  - stahuje se ze sítě (streaming)
    - spustí se, použije a zahodí

The screenshot shows the Microsoft PowerPoint Online (formerly Web App) interface running in a web browser. The slide content is identical to the desktop version. A yellow callout box with the same text "„Click to Run“ řešení á la tenký klient (NC)" points to the slide content. The browser address bar shows the URL for Microsoft OneDrive.

## • verze Online (dříve Web App)

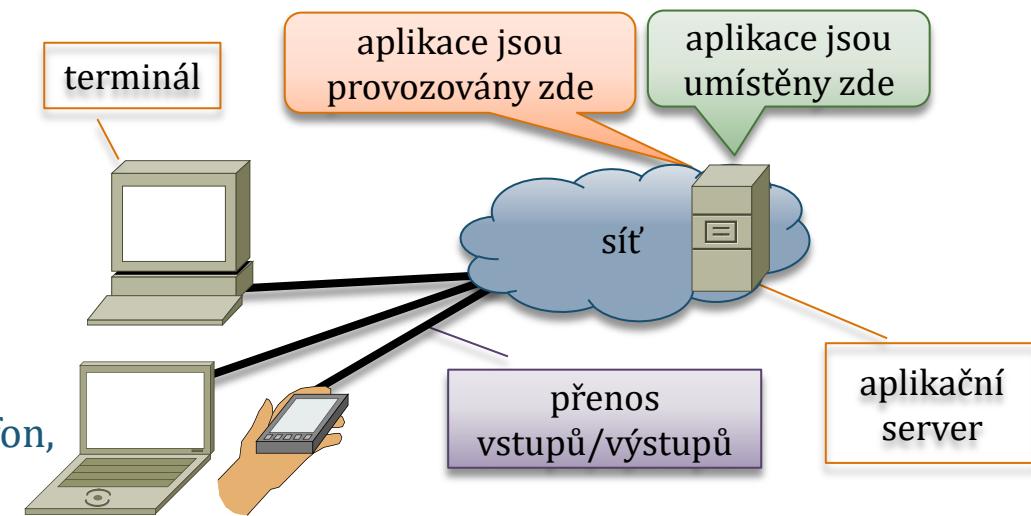
- forma RIA (Rich Internet Application)
  - napsáno v Javascriptu
  - běží v rámci browsoru

# Server-Based Computing

- samostatný výpočetní model
  - lze chápat též jako dílčí variantu modelu „Network-centric computing“
- podstata:
  - celá aplikace běží v síti, na vhodném serveru
    - na tzv. **aplikačním serveru** (obdoba hostitelského počítače)
      - který jako svou službu poskytuje možnost provozování aplikací
  - koncové zařízení (u uživatele) se chová jako **terminál**
    - jsou k němu přenášeny pouze:
      - výstupy aplikace
        - ale již v **plné (rastrové) grafice**, lze mít plně grafické GUI
      - vstupy od uživatele
        - stisky kláves, pohyby myší

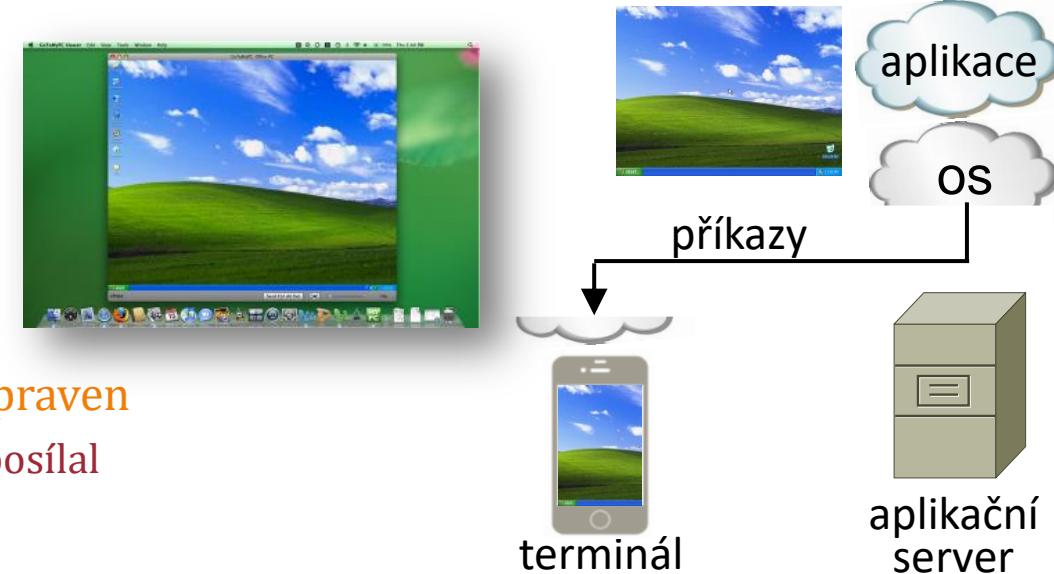
stejné jako u modelu host/terminál

rozdíl oproti modelu host/terminál



# Server-Based Computing

- problém modelu Server-based computing:
  - plně grafická (rastrová, bitmap-ová) data, generovaná aplikací (běžící na aplikačním serveru) jsou stále moc velká
    - než aby bylo únosné je přenášet po síti (do koncového zařízení uživatele)
- řešení:
  - ke generování (rastrových) grafických dat dochází až na koncovém zařízení uživatele
    - a od aplikace přichází pouze příkazy typu „*vykresli okno velikosti XY na souřadnicích AB*“
      - pokud jsou tyto příkazy vhodně navrženy, může být objem přenášených dat minimální
        - kromě potřeby přenosu rastrové grafiky, kde tento přístup selhává
    - představa: grafický subsystém operačního systému se „*vyřízne*“ a přenese do terminálu
      - kde teprve generuje svá data
- důsledek:
  - na terminálu musí být instalována potřebná podpora
    - pro vykreslování v grafice
      - jakýsi „klient“
  - také (aplikační) server musí být upraven
    - aby negeneroval grafická data, ale posílal terminálu příkazy



# příklady: X-Window, XenApp

- na modelu **Server-Based Computing** fungují (například):

- systém **X-Window** (X11, X)

- starší řešení ze světa Unixu, prezentované jako řešení klient/server

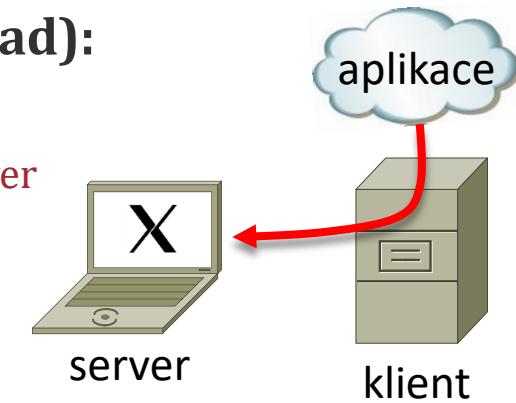
- s „obrácenou“ terminologií:

- aplikace běží na klientovi

- pracovní stanice uživatele je serverem:

- poskytuje aplikaci službu, spočívající v zobrazování

- v generování grafických (rastrových) dat, přímo na uživatelském zařízení



- systém **WinFrame** (**MetaFrame**, dnes **XenApp**) společnosti Citrix, 1995

- původně: upravený server Windows NT 3.51, funguje jako aplikační server

- pro více uživatelů současně

- mohou na něm provozovat běžné aplikace pro Windows

- grafický subsystém přenesen na terminál

- zde musí být instalována potřebná podpora

- WinFrame klient (ICA klient, Online plugin, Citrix Receiver , XenApp plugin)

- pro komunikaci mezi aplikačním serverem a terminálem slouží protokol **ICA**

- ICA: Independent Computing Architecture

- jde o příkazy k vykreslování, které klient na terminálu provádí



# příklady: MS Terminal Services

- na modelu **Server-Based Computing** funguje (například):
  - **Terminal Services** společnosti Microsoft
    - 1997: Microsoft koupil licenci od Citrixu a zakomponoval ji do Windows NT 4.0
    - 1998: výsledkem je aplikacní server MS Terminal Server Edition (code name Hydra)
      - jako samostatný produkt
    - později: funkce aplikacního serveru (MS Terminal Services) zabudovány do „standardních“ serverů Windows
      - do MS Windows Server 2000, do Windows Server 2003
  - **Remote Desktop Services (RDS)** společnosti Microsoft
    - 2009: vznikly přejmenováním z Terminal Services v MS Windows Server 2008
    - součástí je např. Připojení ke vzdálené ploše
      - klient je standardní součástí všech Windows
      - server je součástí vyšších verzí Windows
        - např. Windows 7 Ultimate a vyšších

